NASA
CP
2143
c.1

NASA Conference Publication 2143

LOAN COPY:
AFWL TECHN
KIRTLAND AF
0079331
TECH LIBRARY KAFB, NM

IPAD: Integrated Programs for Aerospace-Vehicle Design

FOR EARLY DOMESTIC DISSEMINATION

Because of its significant early commercial potential, this information, which has been developed under a U.S. Government program, is being disseminated within the United States in advance of general publication. This information may be duplicated and used by the recipient with the express limitation that it not be published. Release of this information to other domestic parties by the recipient shall be made subject to these limitations.

Foreign release may be made only with prior NASA approval and appropriate export licenses. This legend shall be marked on any reproduction of this information in whole or in part.

Review for general release September 30, 1982.

Proceedings of a national symposium
held in Denver, Colorado
September 17-19, 1980

NASA

IPAD

A national resource for increased productivity

ITAB

TECH LIBRARY KAFB, NM
0099331

NASA Conference Publication 2143

IPAD: Integrated Programs for Aerospace-Vehicle Design

Proceedings of a national symposium sponsored by the National Aeronautics and Space Administration and the Industry Technical Advisory Board and held in Denver, Colorado, September 17-19, 1980

NASA
National Aeronautics
and Space Administration

**Scientific and Technical
Information Branch**

1980

PREFACE

To respond to the national needs for improved productivity in engineering design and manufacturing, a NASA supported joint industry/government project is underway denoted Integrated Programs for Aerospace-Vehicle Design (IPAD). The IPAD project objective is to improve engineering productivity through better use of computer-aided design and manufacturing (CAD/CAM) technology. It focuses on development of technology and associated software for integrated company-wide management of engineering information. The IPAD project is carried out primarily through a contract to The Boeing Commerical Airplane Company under the guidance of an Industry Technical Advisory Board (ITAB) composed of representatives of major aerospace and computer companies. Results to date are believed useful to a broad segment of both aerospace and nonaerospace organizations concerned with computer-aided design and manufacturing (CAD/CAM) technology. To broaden awareness of this technology, a summary of IPAD project accomplishments and plans, as well as industry activities to exploit the technology, was presented. Program papers summarized IPAD project work to define a future company-based CAD system, developments underway to address the critical CAD issues of handling company-wide engineering information, aerospace and computer industry experiences and plans for exploiting IPAD technology, and closely coupled IPAD-related activities underway through such efforts as the Air Force Integrated Computer-Aided Manufacturing (ICAM) program. This Symposium was cosponsored by the NASA Langley Research Center and the Industry Technical Advisory Board (ITAB).

This document contains the manuscripts of the presentations and the prepared comments of the panel discussion. Use of trade names or names of manufacturers in this report does not constitute an official endorsement of such products or manufacturers, either expressed or implied, by the National Aeronautics and Space Administration.

INDUSTRY TECHNICAL ADVISORY BOARD (ITAB)

SYMPOSIUM STEERING COMMITTEE

M. P. Anderson
Ford Motor Company

Len Beitch
General Electric Company

Cliff Enault
Boeing Commercial Airplane Company

Robert E. Fulton
Langley Research Center

Charles Hall
Martin Marietta Corporation

C. R. Lewis
General Motors Technical Center

Si Lyle
Digital Equipment Corporation

Robert R. McCready
Vought Corporation

Ralph E. Miller, Jr.
Boeing Commercial Airplane Company

George C. Salley
Langley Research Center

Warren E. Swanson
Playa del Rey, CA

CONTENTS

PREFACE iii

1. THE IMPORTANCE OF COMPUTERS IN AEROSPACE TECHNOLOGY
 (Paper not received in time for publication)
 Walter Olstad

2. PRODUCTIVITY: CHALLENGE OF THE FUTURE (Paper not received
 in time for publication)
 Norman Augustine

3. IPAD PROJECT OVERVIEW 1
 Robert E. Fulton

4. INDUSTRY INVOLVEMENT IN IPAD THROUGH THE INDUSTRY TECHNICAL
 ADVISORY BOARD 21
 Warren E. Swanson

SESSION I - A FUTURE INTEGRATED CAPABILITY (FULL IPAD)
Chairman: Dan Weigel

5. INTEGRATION OF DESIGN INFORMATION 27
 Gordon L. Anderton

6. FUTURE INTEGRATED DESIGN PROCESS 43
 Donald D. Meyer

7. REQUIREMENTS FOR COMPANY-WIDE MANAGEMENT OF ENGINEERING
 INFORMATION 59
 John W. Southall

8. PRELIMINARY DESIGN OF A FUTURE INTEGRATED DESIGN SYSTEM 75
 Ralph M. Diggins

9. AN APPROACH TO ENGINEERING SOFTWARE (Paper not received
 in time for publication)
 David Loendorf and James Browne

SESSION II - DEVELOPMENT OF AN ENGINEERING DATA MANAGEMENT
CAPABILITY (FIRST-LEVEL IPAD)
Chairman: Stig Wahlstrom

10. EXECUTIVE AND COMMUNICATIONS SERVICES TO SUPPORT THE
 IPAD ENVIRONMENT 95
 J. G. Tanner, D. M. Kirkwood, and F. M. Ives

11. AN ENGINEERING DATA MANAGEMENT SYSTEM FOR IPAD 145
H. R. Johnson, D. L. Comfort, and D. D. Shull

12. AN APPROACH FOR MANAGEMENT OF GEOMETRY DATA 179
R. Peter Dube, Gray J. Herron, Jean E. Schweitzer, and Erich R. Warkentine

13. USER INVOLVEMENT IN IPAD SOFTWARE DEVELOPMENT 203
Walter A. Bryant and Harold A. Crowell

14. IPAD PRODUCTS AND IMPLICATIONS FOR THE FUTURE 219
Ralph E. Miller, Jr.

SESSION III - INDUSTRY APPROACH TO EVALUATION AND USE OF IPAD
Chairman: Len Beitch

15. CASE STUDY - LOCKHEED-GEORGIA COMPANY INTEGRATED DESIGN PROCESS 235
C. Thurman Waldrop

16. INTEGRATED CAD/CAM: PROBLEMS, PROGNOSIS, AND ROLE OF IPAD 247
Edwin N. Nilson

17. COMPUTER-AIDED DESIGN DEVELOPMENT TRANSITION FOR IPAD ENVIRONMENT 259
Herschel G. Owens, Wendell D. Mock, and James C. Mitchell

18. IPAD: A COMPUTER VENDORS PERSPECTIVE 279
Harley D. Feldman

19. IMPLEMENTING AN INTEGRATED ENGINEERING DATA BASE SYSTEM - A DEVELOPER'S EXPERIENCE AND THE APPLICATION TO IPAD 285
Everett A. Bruce

20. APPLIED GRAPHICS IN ENGINEERING DESIGN (Paper not received in time for publication)
Robert Simek

LUNCHEON SPEAKER

21. IPAD: AS IT RELATES TO THE UTILIZATION OF CAD/CAM IN GENERAL MOTORS - SYNOPSIS 295
Betsy Ancker-Johnson

SESSION IV - PANEL DISCUSSION ON USE OF CAD SYSTEMS AND RELATIONSHIPS TO IPAD
Chairman: Si Lyle

22. TURNKEY CAD/CAM SELECTION AND EVALUATION 299
Ted Moody

23. OBSERVATIONS BASED ON DEVELOPMENT OF A COMPUTER AIDED DESIGN SYSTEM 305
Howard Best

24. INTERACTIVE GRAPHICS 311
Dale Christensen

25. DATA BASE SYSTEMS IN ELECTRONIC DESIGN ENGINEERING 317
Del Williams

26. SYSTEMS ARCHITECTURE FOR DISTRIBUTED APPLICATIONS 325
Richard Peebles

27. A REMOTE COMPUTER GRAPHICS USER AT GENERAL MOTORS 341
Harry S. Murphy

SESSION V - ONGOING IPAD-RELATED ACTIVITIES
Chairman: George Salley

28. ICAM APPROACH TO DESIGN MANUFACTURING INTERACTIONS
(Paper not received in time for publication)
Robert Schafrik

29. MANUFACTURING COST/DESIGN SYSTEM: A CAD/CAM DIALOGUE 353
Henry H. Loshigian, Bernard I. Rachowitz, and David Judson

30. IMPLEMENTATION OF IPAD SOFTWARE IN A UNIVERSITY ENVIRONMENT
(Paper not received in time for publication)
Michael Wozny and Larry Feeser

31. TURNKEY CAD/CAM SYSTEMS' INTEGRATION WITH IPAD SYSTEMS 369
Robert E. Blauth

32. INTEGRATION: A COMPETITIVE EDGE (Paper not received in time for publication)
James Hardy

GLOSSARY 379

IPAD PROJECT OVERVIEW

Dr. Robert E. Fulton
NASA Langley Research Center
Hampton, VA

SUMMARY

To respond to national needs for improved productivity in engineering design and manufacturing, a NASA supported joint industry/government project is underway denoted Integrated Programs for Aerospace-Vehicle Design (IPAD). The objective is to improve engineering productivity through better use of computer technology. It focuses on development of technology and associated software for integrated company-wide management of engineering information. The project has been underway since 1976 under the guidance of an Industry Technical Advisory Board (ITAB) composed of representatives of major engineering and computer companies and in close collaboration with the Air Force Integrated Computer-Aided Manufacturing (ICAM) program. Results to date on the IPAD project include an in-depth documentation of a representative design process for a large engineering project, the definition and design of computer-aided design software needed to support that process, and the release of prototype software to integrate selected design functions. Ongoing work concentrates on development of prototype software to manage engineering information, and initial software is nearing release. This paper provides an overview of the IPAD project and summarizes goals, plans, and progress to date.

INTRODUCTION

The national need for improved productivity has become increasingly apparent with recent statistics of zero or negative growth in gross national product. Significant improvements in aerospace productivity are believed possible through effective utilization of current and future CAD/CAM technology (figs. 1 and 2). A joint NASA/industry $15M project, Integrated Programs for Aerospace-Vehicle Design (IPAD), has been underway for several years and is making significant progress in advancing integrated CAD/CAM technology. The project goal is to increase U.S. aerospace industry productivity through application of computers for integrated company-wide management of engineering data.

In the early 1970's, NASA-funded feasibility studies (refs. 1 and 2) showed that dramatic increases in engineering productivity were feasible through the automation of routine information handling tasks. These results, which were extensively reviewed by the aerospace and computer industry (refs. 3 and 4), showed that such automation would directly decrease cost and flow time in the product design process and would improve the competitive position of the U.S. aerospace industry. Based on these and other results, NASA began the IPAD

project in 1976 to develop the appropriate technology and associated computer software. Work under the IPAD project is being done principally through a NASA prime contract to The Boeing Commercial Airplane Company supported by appropriate subcontracts and under the guidance of an Industry Technical Advisory Board (ITAB) composed of members of aerospace and computer companies (fig. 3). The ITAB concept provides an innovative and effective management approach for a joint industry/government high technology R&D effort (ref. 5). (See Appendix A for a summary of industry involvement in IPAD).

IPAD software development is closely coordinated with the U.S. Air Force Integrated Computer-Aided Manufacturing (ICAM) program, and several IPAD/ICAM cooperative activities are underway or planned. (See, for example, refs. 6 and 7). NASA management of software development is supported by continuing independent evaluation of software performance carried out through a contract to Information Research Associates (ref. 8). While IPAD software is being developed primarily for use by the aerospace industry, it should be useful in support of other complex processes such as large civil engineering projects, shipbuilding, automotive design, electronics, and engineering education (refs. 9, 10-13). Representatives from several non-aerospace organizations serve as observers to ITAB and regularly participate in the review and critique of ongoing work (fig. 3).

DEVELOPMENT PLAN

A fully operational IPAD capability of the future (fig. 4) would be composed of system software including executive, data management, and geometry/graphics software, together with disciplinary technical programs installed in IPAD to implement its integrated design features and project data. The IPAD project focuses on developing prototype system software and would use available technical programs and data for software evaluation and demonstration. It would also take advantage of the capabilities of the host computer interactive operating system.

Major IPAD project tasks during FY 1976-82 (see figs. 4 and 5) include (1) definition of a computer-aided design system of the future denoted Full IPAD; and (2) development of First-Level IPAD software principally composed of a prototype engineering data management software system operational on CDC and IBM computers supported by geometry and graphics software operational on a DEC VAX minicomputer. IPAD software development is being carried out through a careful step-by-step process encompassing aerospace design process definition, system requirements definition, software design, and software development (fig. 6). The schedule in figure 5 indicates that the Full IPAD design has been completed; work is well into the CDC development with geometry/graphics software released; data management software is nearing release; and planning for IBM implementation is underway. The following sections provide further details on the description of a future Full IPAD, as well as on progress and plans toward developing a First-Level IPAD software capability.

DESCRIPTION OF A FUTURE FULL IPAD SYSTEM

IPAD project results to date (see refs. 14-24) indicate that a Full IPAD system of the future should be basically a general-purpose interactive computer-aided design system developed to support engineering design processes. Its primary function would be to handle engineering data associated with the design process. IPAD software would be installed by each company on its computers and used in a manner similar to vendor-supplied operating system software. The IPAD software would augment, rather than replace, existing operating system software. It would support the continuous design activities of a typical company mix of multiple development projects. The IPAD system would serve management and engineering staffs at all levels of design (conceptual, preliminary, and final) and aid in the assembly and organization of design data for manufacturing processes.

The IPAD system would support generation, storage, and management of large quantities of data. Its capacity would only be limited by the computer hardware configurations selected by each company. The system would be used in a distributed computing environment having one or more central host computing systems and many remote computing systems. One such arrangement of Full IPAD components is given in figure 7. The number of terminals might be several hundred or more and may be distributed across the host and remote systems. The IPAD software would function on the computer complexes in use today by aerospace corporations, but would achieve its full potential on the computers in the next decade.

The functions of the three major IPAD software components illustrated in figures 4 and 7 are (1) executive software to control user-directed processes through interactive interfaces with a large number of terminals in simultaneous use by engineering and management personnel and to provide communications between computer hardware within and outside the IPAD distributed computing system; (2) data management software to provide a comprehensive, versatile capability for efficiently storing, tracking, protecting, and retrieving exceptionally large quantities of data maintained on multiple storage devices; and (3) geometry and graphics utility software to provide a wide range of capabilities for information and geometry creation, manipulation and display functions including design/drafting and interactive and display graphics.

Libraries within the data bases might include analysis/design computer programs utilized by various disciplinary specialists and extensive quantities of data. The analysis/design computer programs would not be part of IPAD, but would be provided by each company to form the complete design-software system; selected publically available technical programs might be included in IPAD releases to demonstrate capabilities. The data in the data base would include all official project information defining the characteristics of current baselines and alternative designs and their performance, as well as archival "handbook" information forming the technology base for company designs. Simultaneous access to the same baseline design information by all disciplinary groups would thus be possible. Temporary storage for design information being actively used by individuals or teams would also be provided.

A Full IPAD system would not be a hands-off "automated design" system and would not constrain company design methods. The quality of future aerospace designs generated in an IPAD environment would depend on the same primary factors as in today's design environment: creativity of designers, quality of technical staff, quality of analysis tools and design data, and coordination of design and manufacturing information. IPAD should also be a tool to improve manufacturing direct access to engineering data. While support for the manufacturing process is not a specific requirement for the Full IPAD system, it is believed that many manufacturing needs are met by the resulting system design. Continuing collaboration between the Air Force and NASA should insure compatibility between future IPAD and ICAM capabilities so that manufacturing users can more readily take advantage of the design and data management features of IPAD and can have better access to design information.

In summary, the key results to date in defining a future full IPAD system include:

1. Comprehensive description of a future representative aerospace-vehicle design process and its interface to manufacturing (fig. 8, refs. 14-20).

2. Requirements and preliminary design of a future IPAD software system denoted "Full IPAD" to integrate engineering activities of an aerospace company having several products under simultaneous development (fig. 7, refs. 21-24).

FIRST-LEVEL IPAD SOFTWARE DEVELOPMENT

An initial set of IPAD software denoted First-Level IPAD is now under development. This software is a meaningful subset of the Full IPAD system and, in accordance with ITAB guidance, is focusing on critical technology issues associated with managing engineering data. It provides a significant step toward development of a future integrated computer-aided design capability operational on a network of computers of different manufacture. Key requirements driving software development include numerous interactive terminals, voluminous quantities of data (fig. 9), and extensive consideration of geometry (fig. 10).

In progress to date, major technology issues have been addressed, design of a prototype engineering data management system completed, software development progressed substantially, some initial software released, and a major capability nearing release. The result has been identification and implementation of appropriate geometry/graphical software (fig. 11, ref. 25), and the design and substantial progress toward development of an innovative approach to integrated company-wide management of engineering data for a future distributed computer complex (fig. 12, refs. 26-30).

All IPAD technology, software, and supporting documentation is being supplied to industry as it is developed. (See, for example, refs. 30-35). The software development plan provides for release of incremental capabilities implemented for the CDC/DEC and IBM computing systems. These incremental releases will permit each company to incrementally install and evaluate the software and associated technology and, as appropriate, undertake a gradual transition from the current computing environment to a future IPAD integrated environment at a pace appropriate for the company.

During the balance of 1980, work is focusing on coding the engineering data management software and developing demonstrations of integrated design/manufacturing activities to validate the software and its fundamental concepts. An initial engineering data management capability will be operational in late 1980 and future development, together with demonstrations, will continue into mid-1981. Plans for 1981 and beyond are to implement the engineering data management software on a second computer of different manufacture (IBM), to enhance performance of software, to expand software to respond to manufacturing data management requirements, and to examine the technology challenges of computer-aided design/manufacturing on networks of computers (fig. 13). Close collaboration will be maintained with the Air Force ICAM program to insure that the joint IPAD/ICAM programs best respond to the combined technology needs of design and manufacturing.

In summary, First-Level IPAD software produced to date or planned into 1982 include:

1. Release in October 1979 of IPAD Integration Prototype Software operational on a CDC/DEC computer complex which integrates through an experimental relational data management system the functions of design, drafting, geometry, graphical display, structural and thermal analysis/design, and N/C path generation (fig. 11, ref. 25).

2. Release in July 1981 of a prototype instrumented engineering data management system based on a multi-schema concept which is operational on a CDC computer and provides the capability to integrate company-wide design data (fig. 12, refs. 26-30).

3. Release in July 1982 of the prototype engineering data management system operational on an IBM computer complex.

4. Continued documentation of scenarios of representative engineering design activities which serve to control software development, insure requirements are met, and serve as a basis for software demonstration.

APPENDIX A

INDUSTRY INVOLVEMENT IN IPAD DEVELOPMENT

The definition of IPAD has evolved over many years from a study and critique process that included extensive aerospace industry involvement. Two in-depth studies of the feasibility and possible forms of an IPAD system were carried out by The Boeing Company and General Dynamics/Convair (see ref. 1,2). The total cost of these studies over a 17-month period was $611,000. Each study contractor undertook a careful dissection of the vehicle design process to delineate those functions and tasks that can be beneficially supported by computer hardware and software and then defined the format and elements of a software system that could substantially improve the design process. They also assessed the impact of this IPAD system on company computer hardware requirements and on the performance of company staffs and evaluated its cost and benefit potential.

One company examined these questions in the context of design of three kinds of vehicles--a large subsonic transport, a supersonic transport, and a hydrofoil--and developed a comprehensive, detailed picture of the design process as a multilayered network of functions. The other examined intensively the tasks and interfaces of individual designers and groups and analyzed carefully the information flow in design. They considered the effects of the detailed constituent parts of the design process and extrapolated their experience with existing software systems to arrive at computer requirements, costs, and benefits of IPAD software. Both concluded that IPAD is feasible and will fit on existing computers. They arrived at software systems that differed in detail, but exhibited the same general characteristics and order-of-magnitude costs. Projected benefits included 25-90 percent time and 20-60 percent cost savings in design, better management visibility, and reduced risk and cost resulting from greater depth in early trade-offs, on-time designs, and fewer design changes during production.

Results of these studies were presented in four oral reports that were well attended by representatives of industry; for example, 83 industry representatives attended the final oral presentations. Following completion of the studies, the results were critiqued by teams from McDonnell Aircraft Company; Lockheed-Georgia Company; Grumman Aerospace Corporation; Rockwell International, Los Angeles Aircraft Division; Control Data Corporation; IBM Corporation; and Sperry Univac. These firms examined such questions as completeness of the studies, credibility of the proposed systems and projected development parameters, user acceptance, and government and industry roles. They expended significant effort over four months, employing 31 team members and about 100 part-time consultants. The critique reports (ref. 3) reveal a wide spectrum of views, but strong consensus that IPAD development should proceed, should not include technical modules development which should remain largely the prerogative of industry, and should provide early delivery of software and user involvement. Because of the inevitable budget limitations, it was recommended that NASA limit its specific objective to production of a truncated, but "working", system.

Other evaluations of IPAD include an Army-funded study by McDonnell Douglas Astronautics Company of its benefit potential for missile design (ref. 4) and a small NASA-funded study by Battelle Columbus Laboratories of its potential for non-aerospace application (ref. 9). In addition, the NASA Research and Technology Advisory Committee (RTAC) on Materials and Structures sponsored a colloquium of high-level aerospace managers at MIT on January 30-31, 1974, at which IPAD was examined and discussed. NASA prepared an IPAD "Prospectus" in February 1975 which set forth the plan for development, initial maintenance, and release of IPAD; for an Industry Technical Advisory Board (ITAB) to advise the IPAD contractor; and for a user-controlled organization to accept maintenance responsibility for IPAD software. NASA then conducted a survey of 41 aerospace companies seeking their commitment to become a member of ITAB during IPAD development; to evaluate IPAD software before it is generally released; and to financially support, in the context of a user-controlled organization, maintenance and improvement of IPAD software after its value to their company had been demonstrated. Two messages of a general nature were apparent in the company responses. First, support for the IPAD concept and willingness to provide advice and counsel through the ITAB was very good from the large and medium airframe companies for whom IPAD would be primarily tailored. Second, most companies prudently preferred to defer hard commitments beyond ITAB participation until they had a chance to assess results. A few companies specifically declined commitments to participate in the IPAD project, and these fell in two categories - either IPAD did not appear to meet the needs of their particular design process, or they saw IPAD aimed at design problems larger than their company activity. Several such companies wished to remain informed on IPAD progress with an opportunity to re-evaluate their position later.

Based on industry willingness to support the IPAD project, an Industry Technical Advisory Board (ITAB) was formed by the development contractor soon after contract initiation to afford industry the maximum opportunity for influencing the course of IPAD development. ITAB consists of members and observers representing major U.S. aerospace and computer companies and meets periodically. ITAB activities include review of planning and technical documents, critique of key development decisions, ranking of IPAD requirements, identification of demonstration programs, and evaluation of software. As the IPAD software is released, ITAB member companies and other potential IPAD users will be aided in its evaluation and use. More details on ITAB ongoing operation and activities are given in reference 35.

REFERENCES

1. Feasibility Study of an Integrated Program for Aerospace-Vehicle Design (IPAD). NASA CR-132390 through NASA CR-132397, 1973.

2. Feasibility Study of an Integrated Program for Aerospace-Vehicle Design (IPAD). NASA CR-132401 through NASA CR-132406, 1973.

3. Industry Critiques of the Boeing and General Dynamics IPAD Feasibility Study Reports. MDC G5350 (Contract NAS1-12436), McDonnell Douglas Astronautics Company, Apr. 1974.

 - Rosenbaum, Jacob D.: Critique of Boeing and General Dynamics Reports on Feasibility Study of an Integrated Program for Aerospace-Vehicle Design (IPAD). Grumman Aerospace Corp., Feb. 28, 1974.

 - Chasen, S. H.: Industry Critiques of Two IPAD Feasibility Studies. Lockheed-Georgia Co., a Division of Lockheed Aircraft Corp., Feb. 22, 1974.

 - Czysz, Paul: IPAD Critique. McDonnell Douglas Co., McDonnell Douglas Corp., Mar. 18, 1974.

 - Ascani, Leonard: Critique of IPAD Feasibility Studies. Los Angeles Aircraft Div., Rockwell International, Jan. 30, 1974.

 - Feldman, Harley D.: A Critique of Feasibility Studies of an IPAD System. Advanced Systems Div., Control Data Corp., June 1974.

 - Muth, Eugene: IPAD Studies Critique Report. International Business Machines Corp., Mar. 7, 1974.

 - Wilson, William: IPAD Critique. Sperry-Univac, Mar. 26, 1974.

4. Santa, J. E.; and Whiting, T. R.: Application of IPAD to Missile Design. NASA CR-132671, 1974.

5. Swanson, Warren E.: Industry Involvement in IPAD Through the Industry Technical Advisory Board. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 4 of this compilation.)

6. Schafrik, R.: ICAM Approach to Design Manufacturing Interactions. Presented at IPAD National Symposium, Denver, CO, Sept. 17-19, 1980.

7. Loshigian, Henry H.; Rachowitz, Bernard I.; and Judson, David: Manufacturing Cost/Design System: A CAD/CAM Dialog. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 29 of this compilation.)

8. Loendorf, David; and Browne, James: An Approach to Engineering Software Design Evaluation. Presented at IPAD National Symposium, Denver, CO, Sept. 17-19, 1980.

9. A Preliminary Investigation of the Potential Application of the IPAD System to Non-Aerospace Industry. NASA CR-2603, 1975.

10. Simek, Robert: Applied Graphics in Engineering Design. Presented at IPAD National Symposium, Denver, CO, Sept. 17-19, 1980.

11. Wozny, Michael; and Feeser, Larry: Implementation of IPAD Software in a University Environment. Presented at IPAD National Symposium, Denver, CO, Sept. 17-19, 1980.

12. Hardy, James: Integration: A Competitive Edge. Presented at IPAD National Symposium, Denver, CO, Sept. 17-19, 1980.

13. Blauth, Robert E.: Turnkey CAD/CAM Systems' Integration With IPAD Systems. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 31 of this compilation.)

14. IPAD Reference Design Process, NASA CR-2981, 1979.

15. IPAD Product Manufacture Interactions With the Design Process. NASA CR-2982, 1979.

16. IPAD Product Program Management Systems. NASA CR-2983, 1979.

17. IPAD Integrated Information Processing Requirements. NASA CR-2984, 1979.

18. IPAD User Requirements. NASA CR-2985, 1979.

19. Anderton, Gordon L.: Integration of Design Information. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 5 of this compilation.)

20. Meyer, Donald D.: Future Integrated Design Process. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 6 of this compilation.)

21. IPAD Requirements. Document D6-IPAD-70040-D. Contract NAS1-14700 with Boeing Commercial Airplane Company, Dec. 1977.

22. IPAD Preliminary Design. Document D6-IPAD-70036-D. Contract NAS1-14700 with Boeing Commercial Airplane Company, Dec. 1977.

23. Southall, John W.: Requirements for Company-Wide Management of Engineering Information. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 7 of this compilation.)

24. Diggins, Ralph M.: Preliminary Design of a Future Integrated Design System. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 8 of this compilation.)

25. IPAD Integration Prototype Software. NASA Contract NAS1-14700 with Boeing Commercial Airplane Company, Oct. 1979.

26. Tanner, J. G.; Kirkwood, D. M.; and Ives, F. M.: Executive and Communications Services to Support the IPAD Environment. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 10 of this compilation.)

27. Dube, R. Peter; Herron, Gary J.; Schweitzer, Jean E.; and Warkentine, Erich R.: An Approach for Management of Geometry Data. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 12 of this compilation.)

28. Johnson, H. R.; Comfort, D. L.; and Shull, D. D.: An Engineering Data Management System for IPAD. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 11 of this compilation.)

29. Bryant, Walter A.; and Crowell, Harold A.: User Involvement in IPAD Software Development. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 13 of this compilation.)

30. Miller, Ralph E.: IPAD Products and Implications for the Future. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 14 of this compilation.)

31. Waldrop, C. Thurman: Case Study - Lockheed-Georgia Company - Integrated Design Process. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 15 of this compilation.)

32. Owens, Herschel G.; Mock, Wendell D.; and Mitchell, James C.: Computer-Aided Design Development Transition for IPAD Environment. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 17 of this compilation.)

33. Bruce, Everett A.: Implementing an Integrated Engineering Data Base System - A Developer's Experience and the Application to IPAD. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 19 of this compilation.)

34. Nilson, Edwin N.: Integrated CAD/CAM: Problems, Prognosis, and Role of IPAD. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 16 of this compilation.)

35. Feldman, Harley D.: IPAD: A Computer Vendor's Perspective. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 18 of this compilation.)

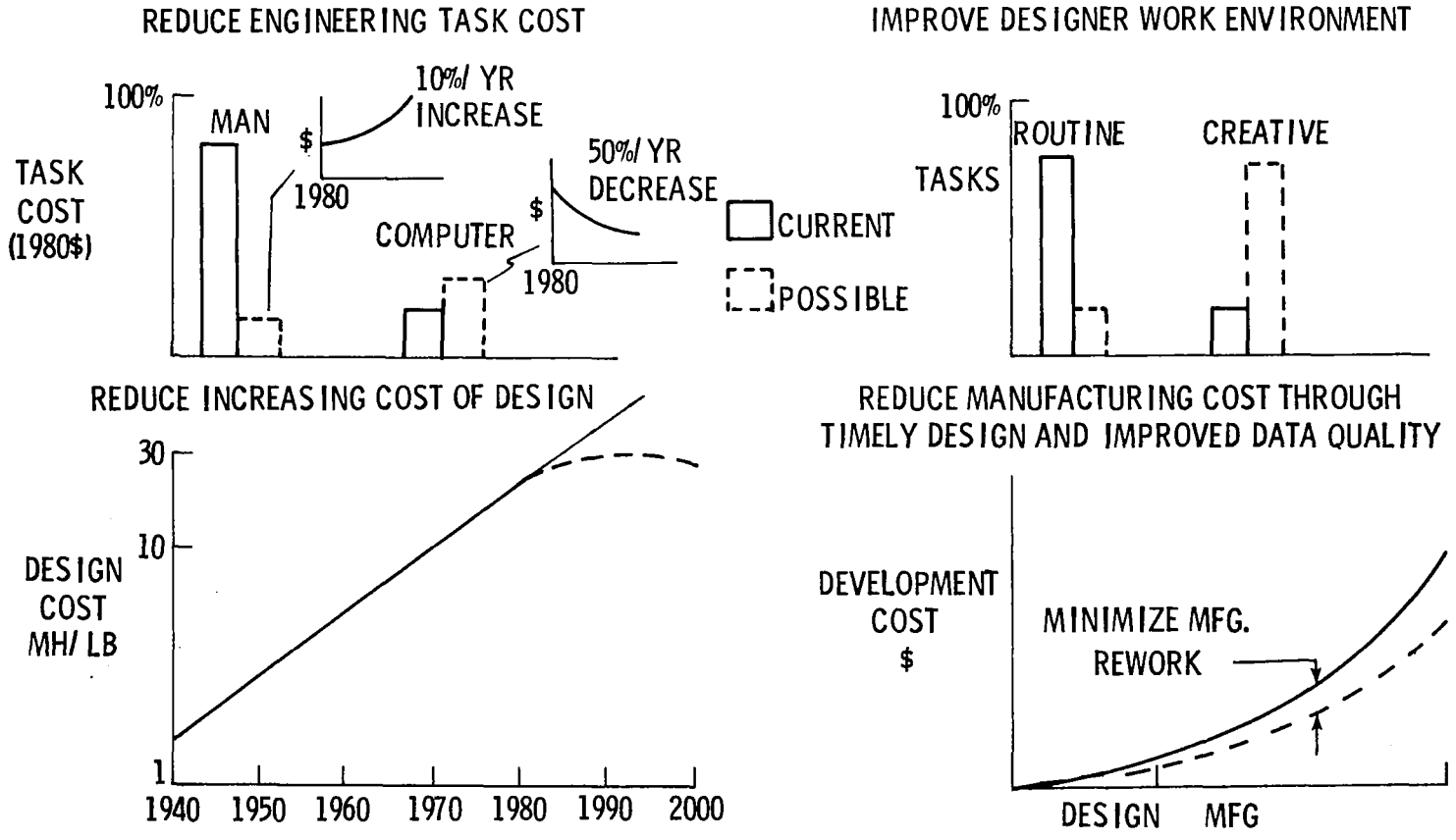


Figure 1.- Benefits from CAD/CAM technology.

GOVERNMENT INVOLVEMENT NEEDED TO ACCELERATE DEVELOPMENT OF NEEDED CAD/CAM TECHNOLOGY

- HIGH RISK LONG-TERM DEVELOPMENT OUTSIDE NORMAL AEROSPACE COMPANY PRODUCT LINE
- HIGH RISK DEVELOPMENT FOR A SMALL COMPUTER COMPANY MARKET
- REQUIRES UNNATURAL COOPERATION AMONG COMPUTER VENDORS
- UTILIZES SCARCE RESOURCES MORE EFFECTIVELY

PROVIDES BASIC TECHNOLOGY NEEDED FOR DOD CAD/CAM PROGRAMS

BENEFIT NASA MISSIONS AND PRODUCTS

Figure 2.- Reasons for NASA development of CAD/CAM technology.

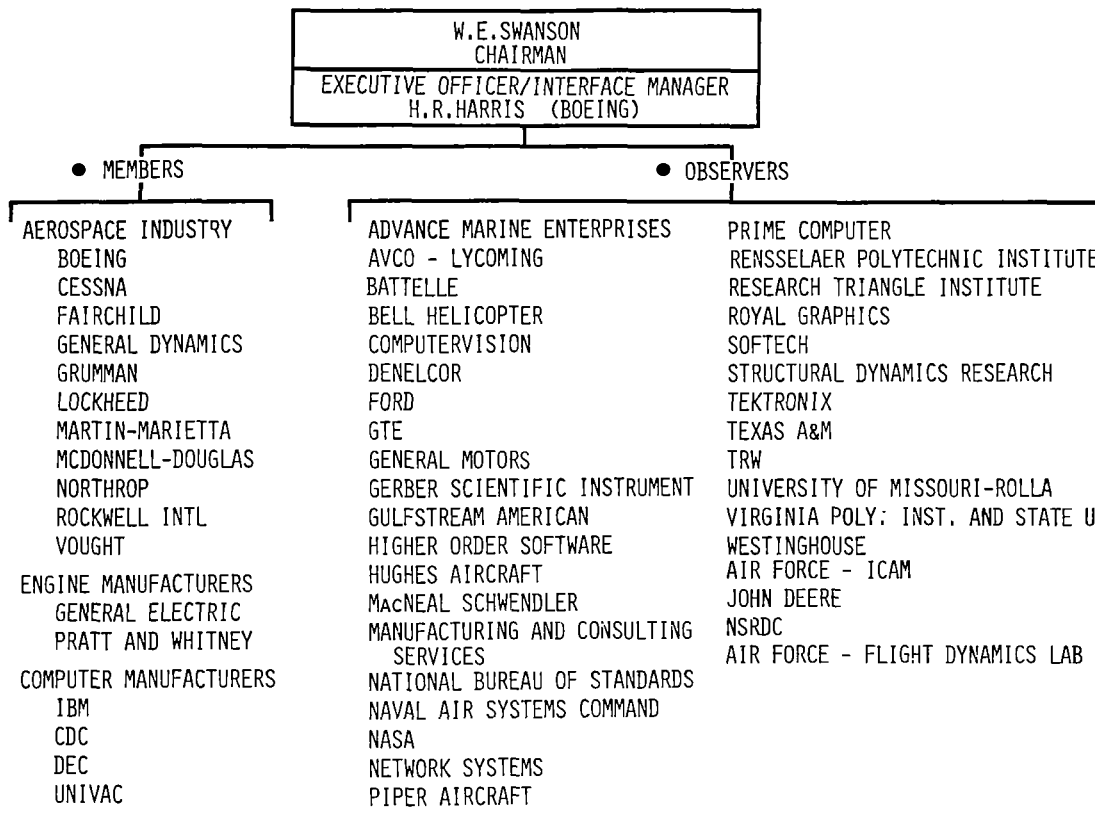
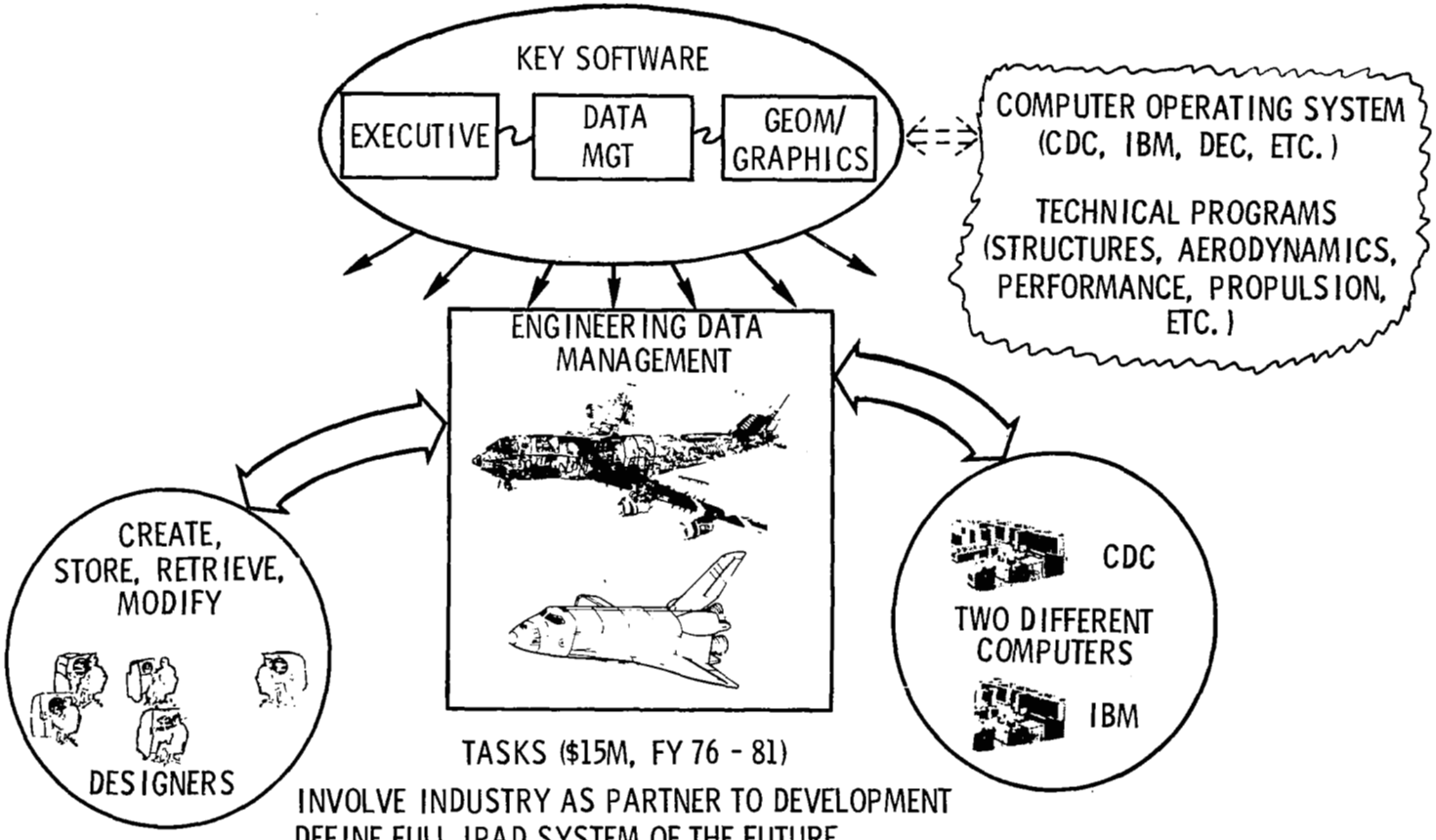


Figure 3.- Industry technical advisory board (ITAB).

GOAL: IMPROVE PRODUCTIVITY THROUGH TECHNOLOGY FOR INTEGRATED, COMPANY-WIDE MANAGEMENT OF DESIGN DATA AND COMPUTER PROGRAMS



TASKS ($15M, FY 76 - 81)

INVOLVE INDUSTRY AS PARTNER TO DEVELOPMENT
DEFINE FULL IPAD SYSTEM OF THE FUTURE
IMPLEMENT BASIC GEOMETRY GRAPHICS CAPABILITY
DEVELOP PROTOTYPE DATA MANAGEMENT SOFTWARE

Figure 4.- IPAD: Integrated Programs for Aerospace-Vehicle Design.

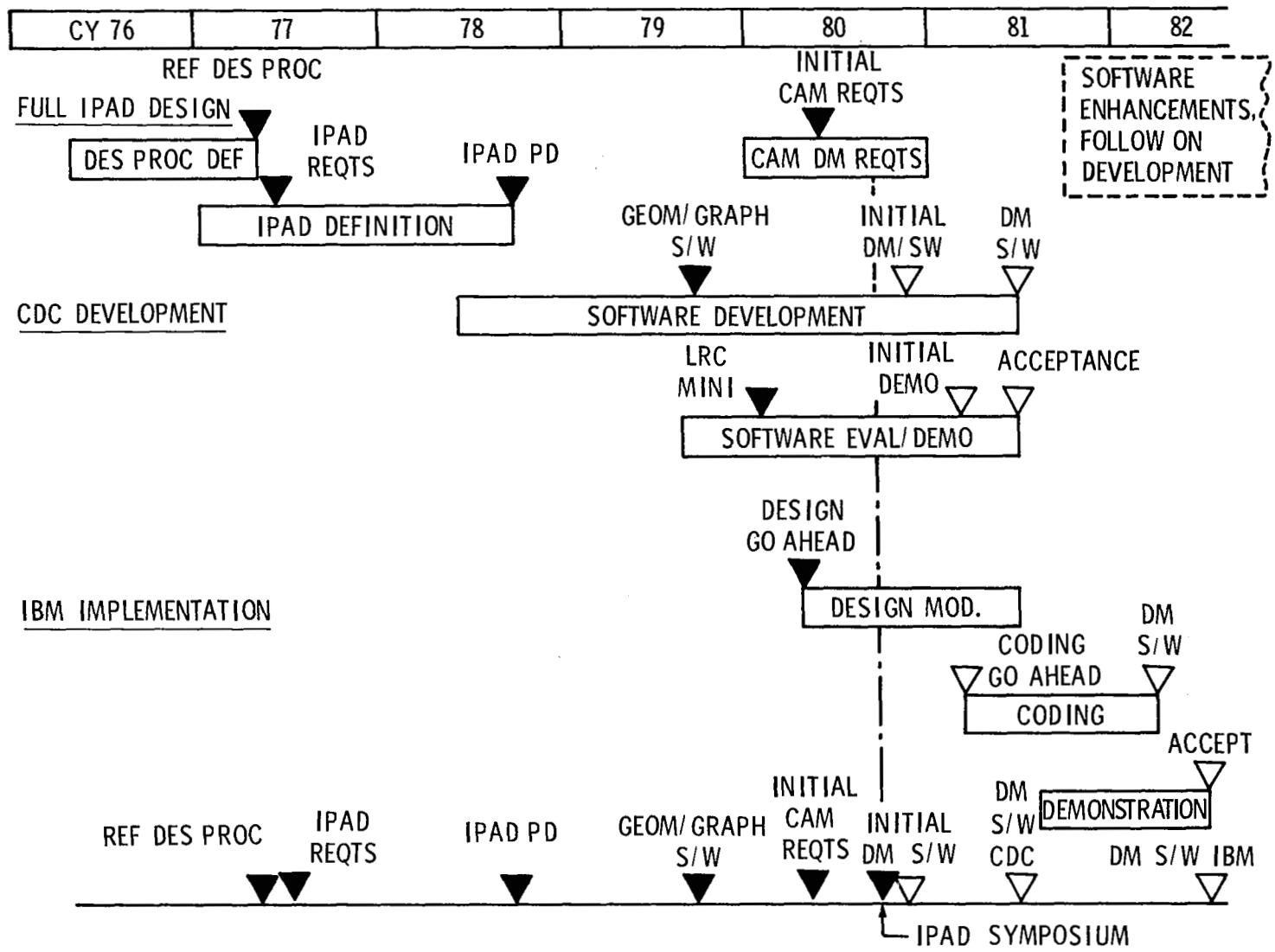


Figure 5.- IPAD development plan (1976-82).

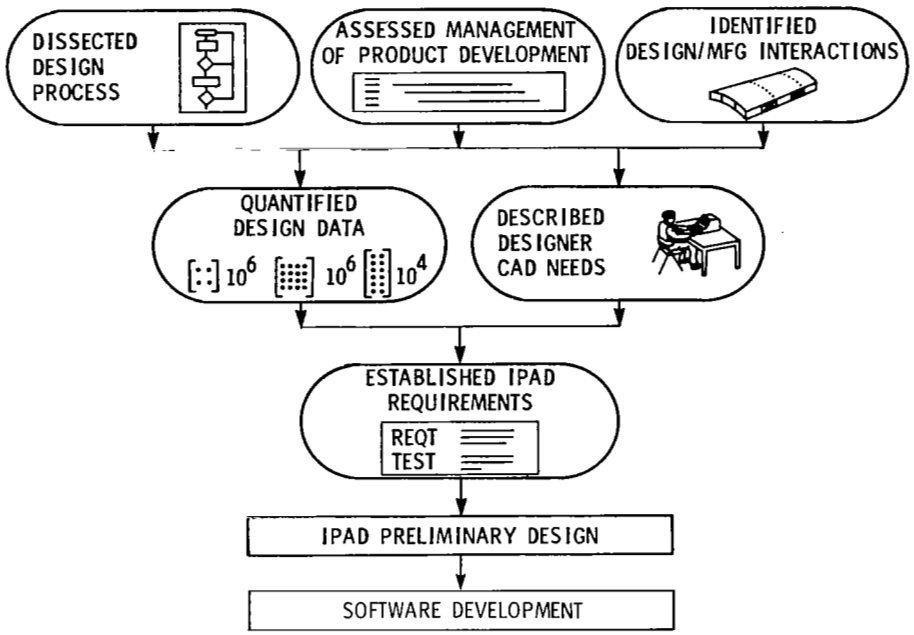


Figure 6.- Approach to IPAD software development.

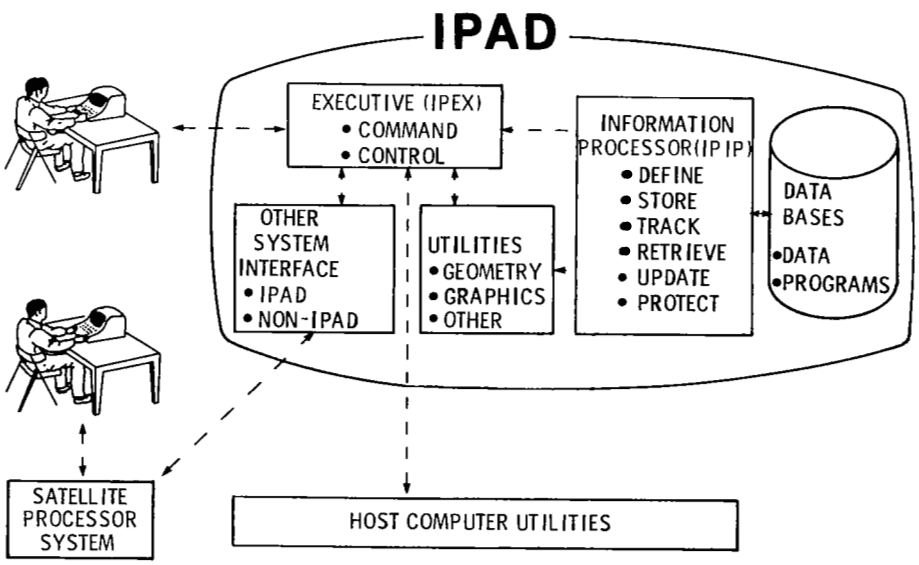


Figure 7.- Arrangement of full IPAD system components.

CTOL, SST, HYDROFOIL VEHICLES

DESIGN LEVELS

I RESEARCH

II–V PRELIM. DESIGN

VI DETAIL DESIGN

VII PRODUCT MFG

VIII PRODUCT VERIFICATION

IX PRODUCT SUPPORT

PART OF STRUCTURAL DETAIL DESIGN NETWORK

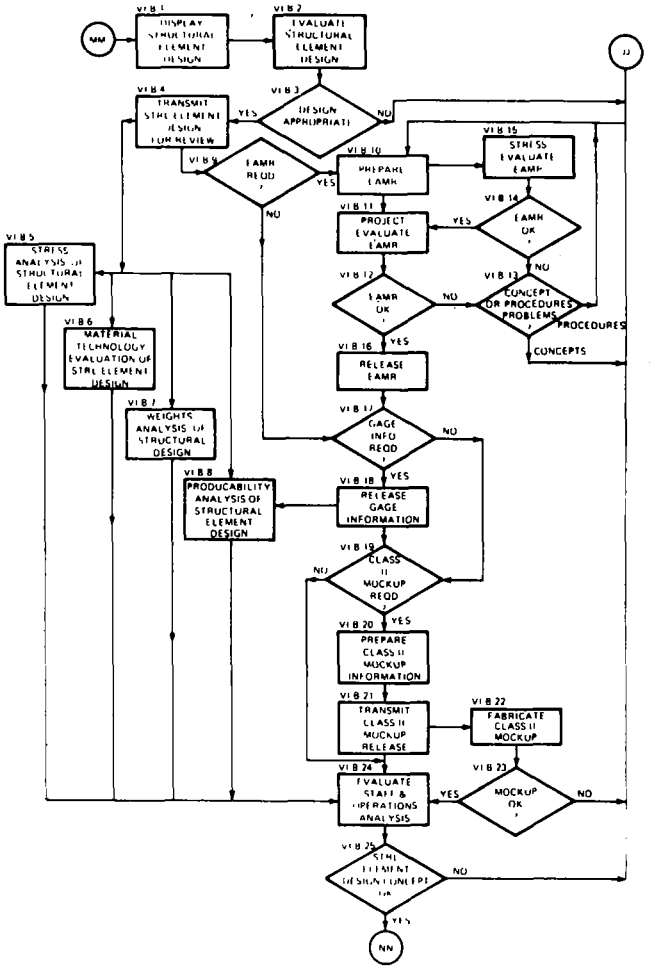


Figure 8.- Detail dissection of design process for representative vehicles.

500 - 1000 SIMULTANEOUS USER TERMINALS

LARGE, RAPID ACCESS DATA VOLUMES TO SUPPORT VEHICLE DESIGN

	ON LINE DATA 10^9 WORDS	10 MINUTE AVAILABILITY 10^9 WORDS
10 PRELIMINARY DESIGNS	1.7	-
10 SUSTAINING DESIGNS	7.8	15.7
2 DETAIL DESIGNS	2.5	3.2
	12.0	18.9

Figure 9.- Key IPAD performance requirements driving software design.

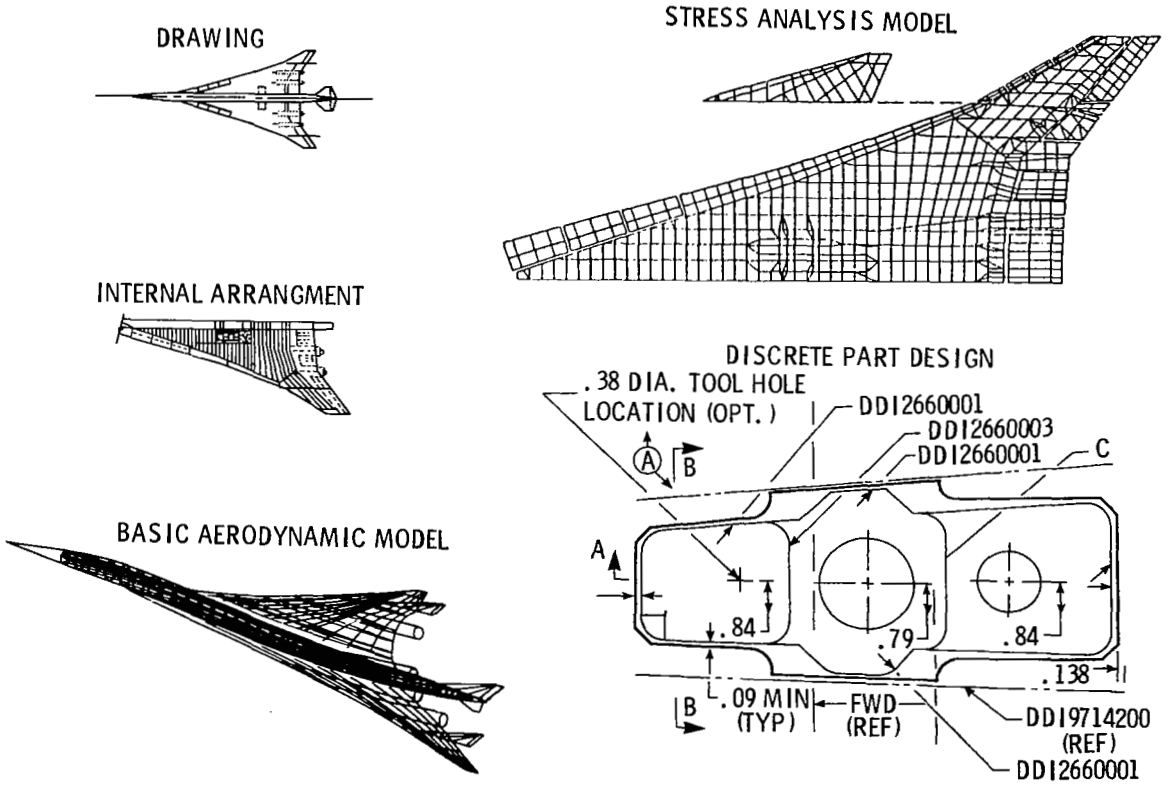


Figure 10.- Geometry permeates the design process.

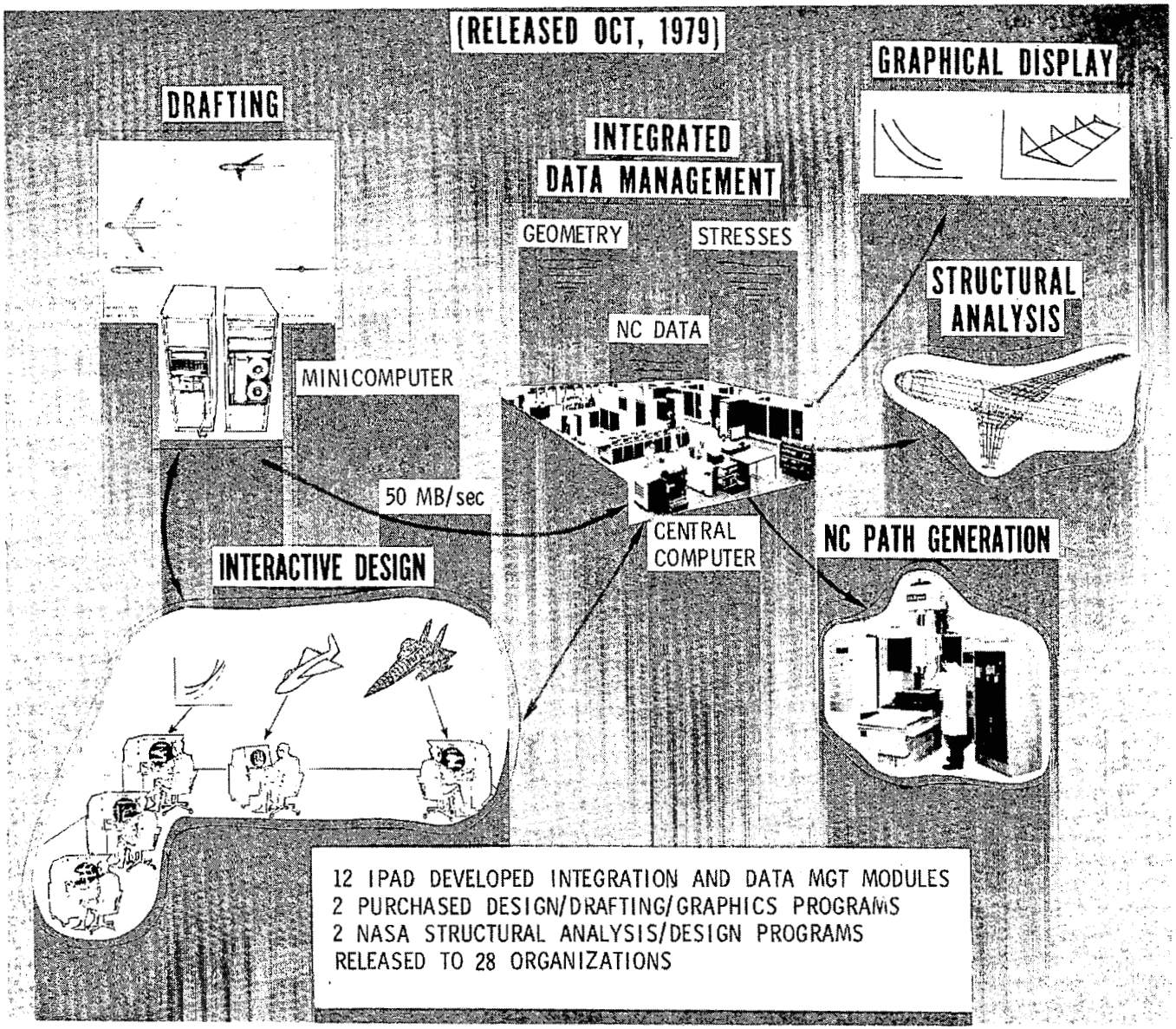


Figure 11.- IPAD design/drafting subsystem.

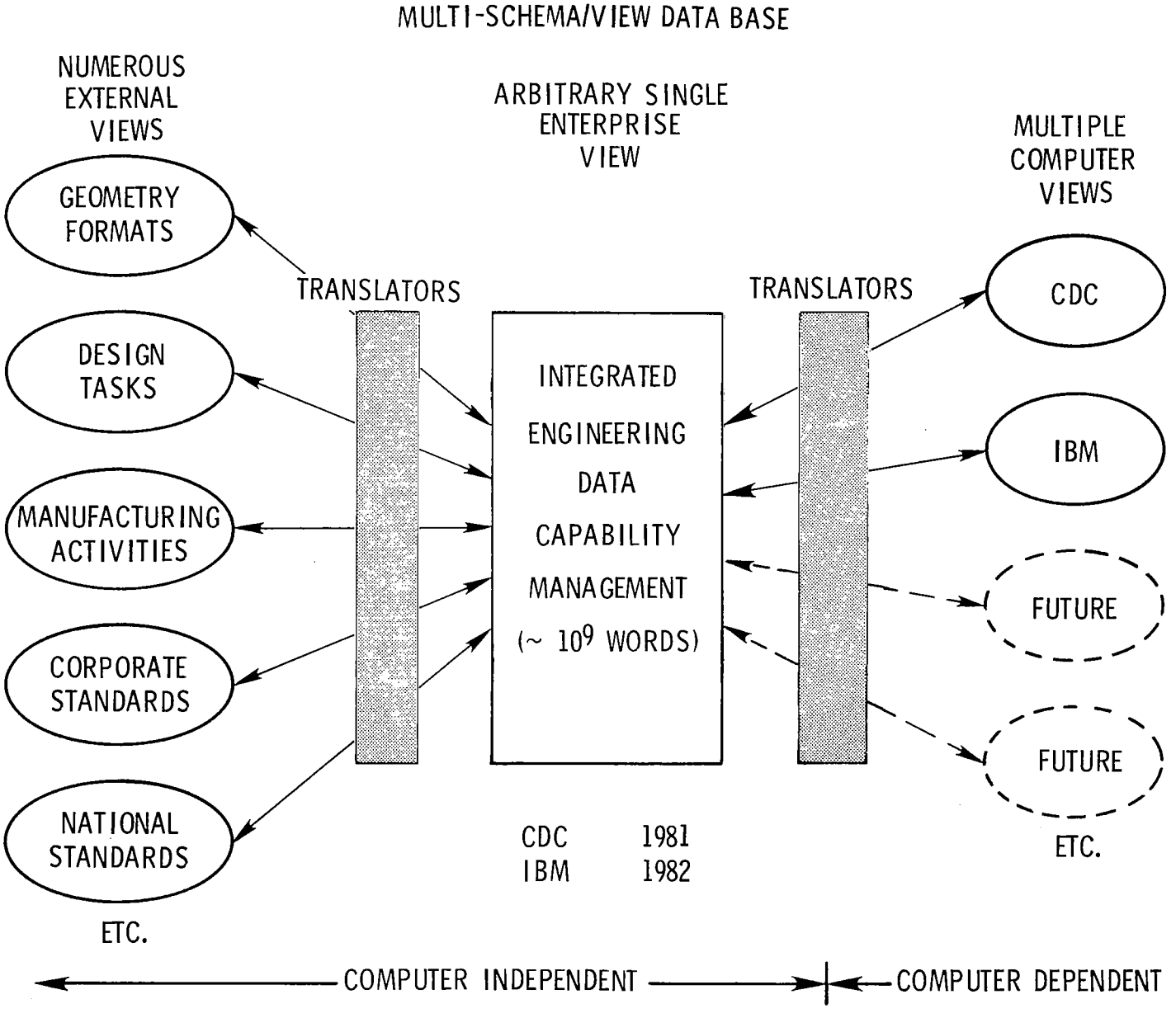


Figure 12.- IPAD engineering data management approach.

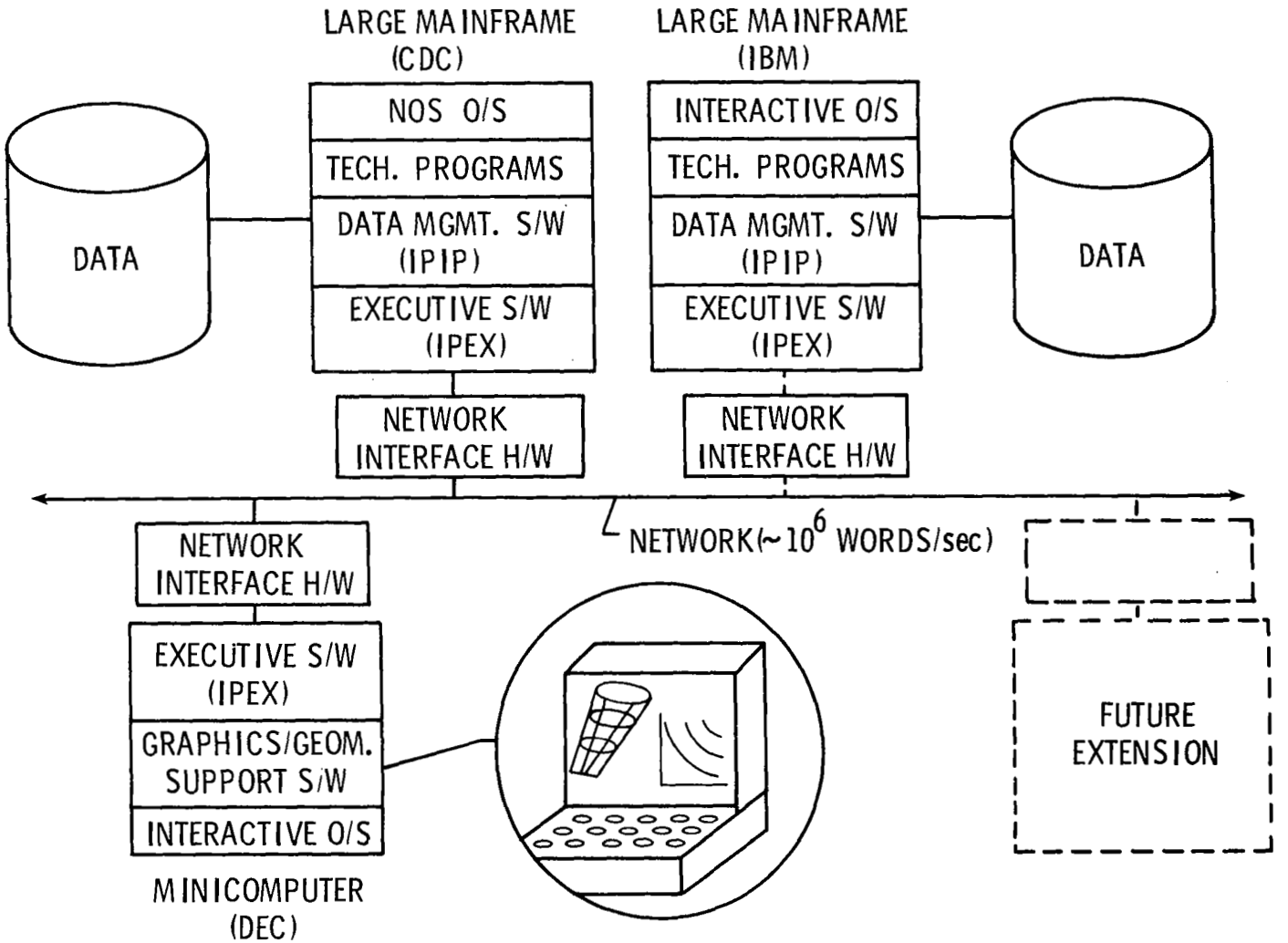


Figure 13.- IPAD hardware/software configuration.

INDUSTRY INVOLVEMENT IN IPAD

THROUGH THE INDUSTRY TECHNICAL ADVISORY BOARD

Warren E. Swanson
Chairman, ITAB

SUMMARY

In 1976 NASA awarded The Boeing Company a contract to develop IPAD (Integrated Programs for Aerospace-Vehicle Design). This contract included a requirement for Boeing to form an Industrial Technical Advisory Board (ITAB), with members representing major aerospace and computer companies. The purpose of this board was to guide the development of IPAD.

The specific goal of IPAD is to increase United States aerospace industry productivity through the application of computers to manage engineering data. This goal clearly is attainable; in fact, IPAD's influence can reach beyond the aerospace industry to many businesses where product development is based on the design-building process. An enhanced IPAD, therefore, is a national asset of significance. The role of ITAB in guiding the development of this system is the subject of this paper.

INTRODUCTION

When the IPAD concept was introduced in the early 1970's, many industry people were skeptical. They envisioned a program that would be too large and costly, in terms of overhead, to be programmable on present-day or even future computers. At the time it seemed that there would be far too much data to be processed efficiently by such a system; for example, an estimate was made that some four billion bits of information would have to be stored and recovered in an interactive atmosphere--an inconceivable achievement. (Although still not entirely feasible, this requirement now appears reasonable in the light of ITAB's continuing review, study, and assignment of priorities.) Another concern was that IPAD would be meaningful only to very large aerospace programs and that individual companies would not be able to operate on an IPAD-type system. There was also some apprehension on the part of industry that an IPAD-type approach might actually dictate the future of the computer industry to suit its own purposes.

In retrospect, it is now clear that industry failed to appreciate fully the IPAD concept, which was to start by

developing potentially feasible data managment requirements on the assumption that concurrent advances being made thoughout the computer industry would produce new computers and storage devices that would eventually support these emerging requirements. As indicated above, this is proving to be the case, largely as a direct result of the relationship between the IPAD program and the Industry Technical Advisory Board.

THE STRUCTURE OF ITAB

The NASA contract with Boeing called for this board to be composed of prospective users of the system, that is, representatives of aerospace and computer companies. Industry's response to the Boeing invitation to serve on the board was amazing: Boeing was besieged with requests for membership. These requests came from almost every segment of these industries and immediately necessitated limiting the total number of members in order to stay within funding limitations. A decision was therefore made to establish a maximum of 20 voting memberships, with support to be provided by 60 to 70 advisory members. This organizational structure seemed to offer the best promise of fulfilling the ITAB charter, which was to provide advice to Boeing, in the development of IPAD, that would be independent of Government direction and separate from The Boeing Company.

ITAB's members have come from data processing companies, computer manufacturing firms, and aerospace manufacturers--both airframe and engine producers. They are industry specialists who are expert in the computing field. Interestingly enough in view of their diversity, many are top-level management personnel.

The structure of ITAB is illustrated in figure 1. The 20 voting members formulate recommendations for IPAD's development after listening carefully to discussion and suggestions from the nonvoting advisors. These recommendations constitute the basic ITAB "product" and ensure a continuous flow of informed guidance to Boeing. It is the interplay of these ITAB recommendations and Boeing responses that determine the program's direction. The communication channels for the flow of ITAB recommendations are depicted in figure 2.

BENEFITS GAINED FROM THE IPAD-ITAB RELATIONSHIP

From the outset, the contractual basis of ITAB ensured vigorous industry participation, for it was evident that industry would be the free recipient of any useful information that might be generated by the IPAD program. Fears about the future being dictated by IPAD gradually diminished as the computer companies

began to see the influence that their representatives were exerting on the IPAD program as well as the value of the information being received in return.

Early in the study phase of IPAD's development, a set of requirements was established as a baseline reference for the design process. It was interesting to note that this design process development could be used by all of the ITAB member companies, including the observers, and that it has since proved helpful to them in their various efforts to develop suitable equipment to meet the demands of emerging requirements. Although agreement on appropriate processes to meet these requirements has by no means been unanimous, there has developed a widespread appreciation of the techniques that have evolved for putting the design process on paper. As a result, each company has been able to take this formatted design and lay it out in a manner that suits its own unique circumstances. This aspect of the ITAB experience alone has been of national benefit in simply describing for a particular company how to go about defining its design process. ITAB involvement is shown in figure 3.

It has also become clear that IPAD is capable of developing a single data management system adaptable to the needs of many if not all companies, thereby offering them efficient data management procedures without the necessity of individually developing such systems. The advantages of this capability are obvious to any manager who has ever struggled with the data management problem.

During the past few years there has been a growing awareness of a dangerous reduction in American industrial productivity. In many U.S. factories productivity has decreased significantly, especially when compared with that of Japan and West Germany. In the course of ITAB meetings, participating members have come to appreciate that IPAD can exert a substantial influence on the improvement of productivity. Some studies show a ten-to-one improvement in producing engineering data; others demonstrate that using companies should experience a minimum throughput improvement of six to one. Although IPAD has not yet reached its full development or demonstrated its future potential, these projections are verifiable, and productivity improvement through the use of IPAD principles can become of great benefit to the nation.

CONCLUDING REMARKS

Attendance at ITAB meetings has been consistent over the years, and participation has been enthusiastic and effective. Nonvoting advisors, many of them from companies for which IPAD was not specifically designed, have shown appreciation for the potential benefits of the system for all of American industry and

have made many important contributions through their advice and suggestions. Twelve meetings of the Industry Technical Advisory Board have been conducted over the years, and from these meetings 160 recommendations have been submitted to Boeing. Of this number, only 25 remain as open items to which Boeing has not yet made response. As a result of these meetings and through the exchange of communications such as are afforded by this symposium, a broader knowledge of IPAD and its resources for improving national productivity are certain to emerge.

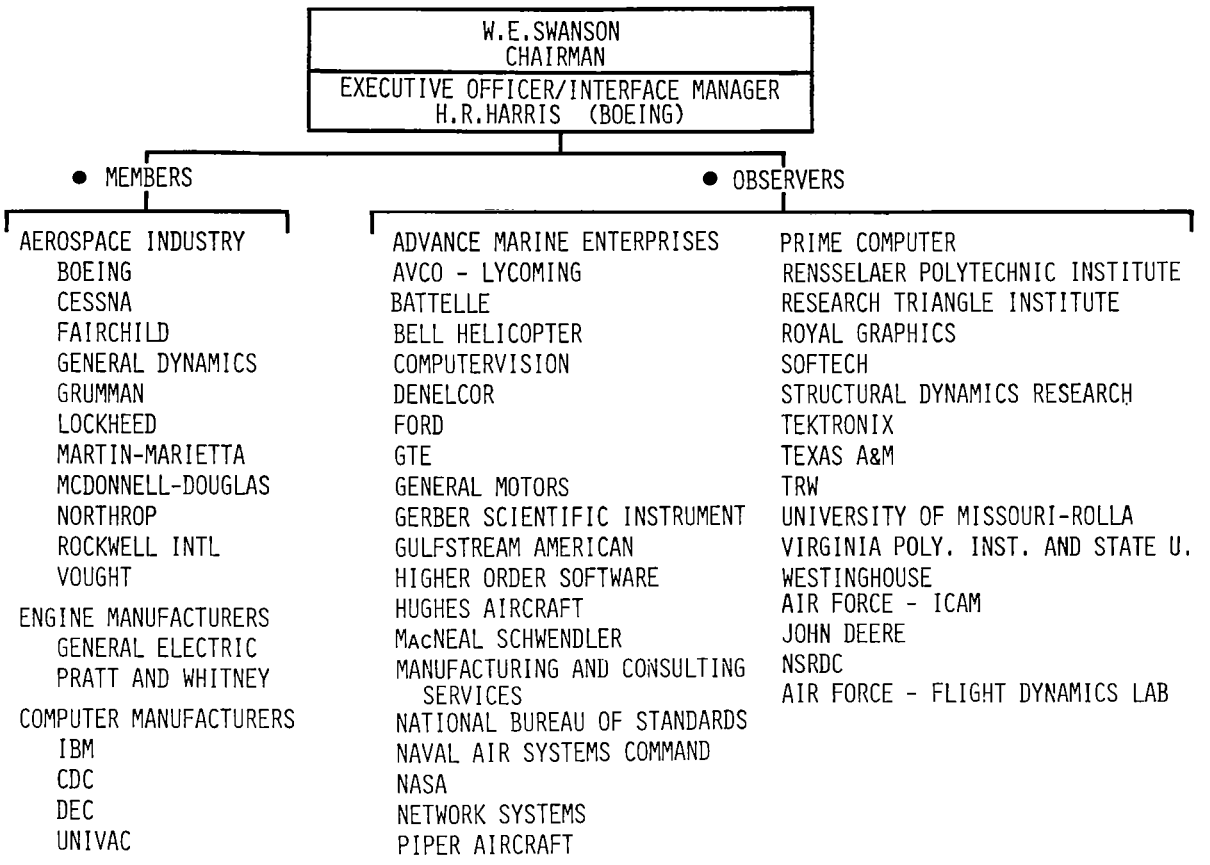


Figure 1.- Industry Technical Advisory Board (ITAB).

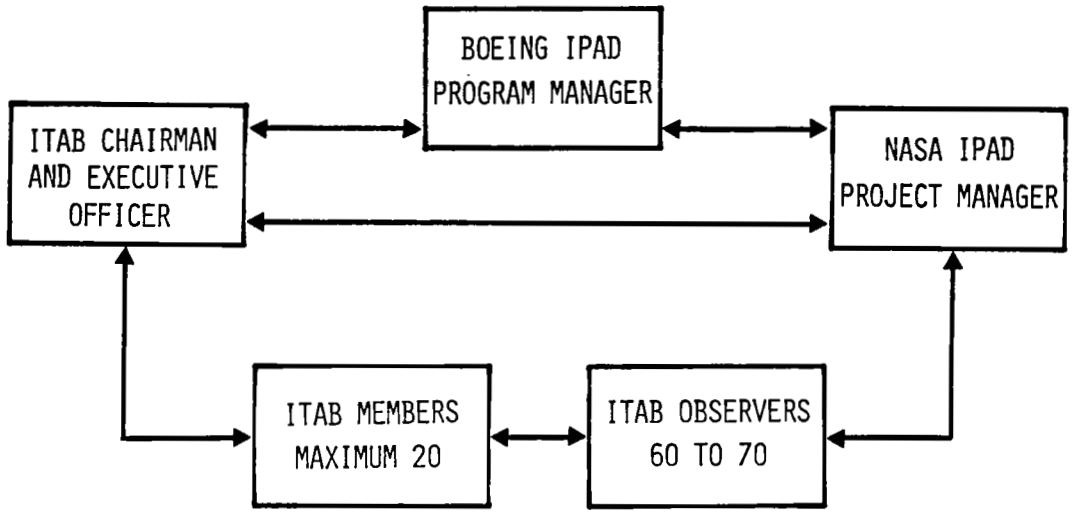


Figure 2.- ITAB organization and communication channel.

- CONTINUING INDUSTRY HIGH-LEVEL CONTRIBUTION AND CONCERN
- PROVIDES OVERALL INDUSTRY INPUTS THEREBY AVOIDING BOEING OR NASA BIASES
- TWELVE REGULARLY SCHEDULED MEETINGS TO DATE
- ATTENDANCE OF 60 - 90 PEOPLE
- ACCOMPLISHMENTS
 - SPECIFIC SUBCOMMITTEES TO ASSESS MAJOR IPAD TASKS
 - APPLICATIONS
 - DEMONSTRATIONS
 - DESIGN/MANUFACTURING INTERACTION
 - NATIONAL SYMPOSIUM
 - FOLLOW-ON CONTRACTUAL TASKS
 - HOSTING AND ATTENDING MEETINGS
 - ON-SITE REVIEW TEAMS
 - REFERENCE DESIGN PROCESS
 - USER REQUIREMENTS
 - FULL-IPAD PRELIMINARY DESIGN
 - DATA BASE MANAGEMENT DESIGN APPROACH

Figure 3.- ITAB involvement.

INTEGRATION OF DESIGN INFORMATION

Gordon L. Anderton
Boeing Commercial Airplane Company

SUMMARY

This paper discusses the overall concepts of the integrated programs for aerospace-vehicle design (IPAD) from the user's viewpoint, provides a top-level view of what the user requires from such a system, and describes the interactions between the system and user. The four major components discussed are design process; data storage, management and manipulation; user interface; and project management.

Although an outgrowth of aerospace production experience, the basic concepts discussed--and especially their emphasis on integration--are considered applicable to all problem solving. Thus, these concepts may offer a broad base for exploitation by industry in general.

This is the first in a set of three papers, the other two being "Future Integrated Design Process," by D. D. Meyer, and "Requirements for Company-Wide Management of Engineering Information," by J. W. Southall. In addition to tying the three together, this paper will discuss in detail how project management can be handled in a computing environment and also the user interface needs.

INTRODUCTION

The Integrated Programs for Aerospace-Vehicle Design (IPAD) is a computer-based system designed to assist the engineer in performing his tasks. It is based on five years' experience with the NASA-funded IPAD development contract. The principal underlying feature of this system is integration: integration of data and application programs, integration of application programs with each other by utilizing design process definitions, and integration of computing jobs with project management (cost and schedules).

The second feature of the IPAD system is the user interface and user assistance. These include uniformity of user language, assistance in using the system, machine independence, special utilities for frequently used applications, geometry definition and creation, and graphics.

The third feature of the IPAD system is information management through an engineering and manufacturing-oriented data base, data integrity and security, data manipulation and query, and application program library.

This is the first in a set of three papers which will describe IPAD from the point of view of the user. This paper presents an overview of the integration characteristics together with a detailed description of the project management and user interface requirements.

ENGINEERING DATA CHARACTERISTICS

Engineering design of a product can be described as locating, creating, modifying, reporting, storing, controlling, and protecting the information that describes the end product, its functional environment, and its performance boundaries. The character of the information is varied and may be analytic or geometric, textual, numeric, or graphic; and it may be either structured or unstructured.

Locating data occupies a large portion of an engineer's time. It is complicated by consideration of the data's currency and quality.

The engineer applies certain source data concerning a product to known physical behavior and thus produces engineering design data that defines the end product in geometric, material, and/or functional terms. In IPAD, this activity is labeled "job" (fig. 1).

If all the jobs were in series, linked together like chain links by common data, then total computerized design would be possible. This, however, is not the nature of aerospace-vehicle design, nor is it an intended goal of IPAD. IPAD is a tool aimed at enhancing today's methods through improved data management and through the automatic coupling of computer programs and their associative data. In cases where several activities (which may be performed by computer programs) can be linked for convenience, the automatic coupling of programs can be specified as a linked activity. In such cases, the total activity is still called a job.

The nature of aerospace-vehicle design is better captured by figure 2, which shows end data influencing source data. This necessitates an iterative procedure that usually converges rapidly to meet the applicable design criteria.

This iterative characteristic results from the large number of influencing parameters and the complex nature of their

interactions. At the highest level, the total product is part of this iterative cycle and is broken down into a progressively more detailed and accurate description of the product through different phases such as configuration design, preliminary design, and detail design.

DATA MANAGEMENT

The iterative nature of design imposes some special and hitherto unique requirements on data management. It must be possible to identify data by a unique name plus a version number and to provide qualifying labels relative to data quality. In addition, data-user relationships must be known. The data and its creator must be recognized by the system in order to establish the correct modify and update security. (Only the data owner can modify data.) The data users must be recognizable by the system so that they can be informed when later versions of data are available.

The large number of users and user groups in an aerospace design organization, each with specialized interests in the data, implies a need for breaking down the data base into a number of data bases called data areas. Data management is covered in detail in a companion paper.

DESIGN PROCESS

All jobs combined, whether simply linked or iterative, comprise the work items that produce the product design, see figure 3. The linking together of jobs through the commonality of shared information can be defined as a design process. This definition implies a chronology for performing jobs, multiple path choices involving engineering expertise, and a possible basis for optimizing the design process. A top level view of the design process is covered in a companion paper.

WORK ELEMENTS

The following terms are used in IPAD for describing work items.

Subtask: Work performed by one employee that contributes to the lowest element of work defined by the WBS at which cost accounting is collected, see figure 4.

Task: Work performed by a group of employees that contributes to the lowest level of WBS at which cost accounting is monitored; a sequence of subtasks accomplished by a group and representing a milestone in the project plan, see figure 4.

Work Breakdown Structure (WBS): A structured index to all elements of work and all items produced by a product program, see figure 5.

Subtask and task are methods of breaking work into packages associated with people, and the WBS is a means of breaking work into packages associated with elements or components of the product. The tying together of subtask, task, and WBS provides a structure that permits the integration of project management with job.

PROJECT MANAGEMENT

Project management can be described by the six activities: planning, organizing, staffing, directing, coordinating, and controlling. Component elements of organizing and controlling, essential to the integration process and lending themselves most readily to a computing environment, are the following:

1. Developing a work breakdown structure (WBS) and technical plan to describe work and expected results of work

2. Developing a business plan and scheduling the time placement of resources required to accomplish the technical plan

3. Assigning work and allocating resources

4. Monitoring tasks and subtasks — supporting reports such as resources (budget/actual) late milestones, milestones due the following week or month, etc.

Structuring Project Management Data

As in the design process description the job is the basic building block of project management. Figure 5 illustrates how the building blocks are arranged to support a WBS, which is a tree structure with each lower level describing the product in greater detail. This procedure is terminated at a level where the work required can be performed by a team of people. The work performed by a single employee is called a subtask, and the completion of a subtask is achieved by the completion of a number of jobs.

Budget manhours are estimated and actual manhours are collected at the subtask level and then rolled up to higher WBS levels.

The running of the first jobs in a WBS can be considered the starting schedule event for that WBS. Similarly, the release of the end data for the last job in a WBS can be considered the ending schedule event. Project design is concerned with the information resulting from a job; project management, on the other hand, is concerned with information about that information, which for convenience, will be called header information, see figure 6. In addition, header data will contain other information about the data (e.g., identification of the creator). This information will address such things as scheduled start and end dates.

IPAD Requirements For Project Management

Data and computer-program-related requirements coming under the project management activity are described in detail below. IPAD will provide the means whereby managers can plan the tasks defined by the design process networks.

"Plan" in this sense, includes scheduling, assigning manpower and computer resources to tasks and subtasks, and indicating the dependency of tasks on each other, see figure 3.

The system will provide the means to name subtasks and tasks by project, to address various levels of planning, and to enable the planner to assign task and subtask dependency (design process definition).

The planner will be able to assign budget and estimated manhours, and computer resources to named tasks and subtasks together with information on scheduling. Review of manpower and computer resources, both actual and estimated, will be available (a) at the terminal using the query processor or (b) offline by hardcopy reports. The input will consist of project, WBS, task, or subtask name and a selection of any combination of schedule, manpower, computer resources, and dependency. A method of selecting special display formats (e.g., bar chart or other graphical representations) will be available. A standard set of symbols will be available to denote key events and activities and will include start/complete project; complete key tasks; decision milestone; and begin milestone/end milestone. The planning information will be stored along with the "header" that applies to all data and programs.

To assist in planning work activities, IPAD will provide the means to query the program headers to determine the planning, status, and other information about any subtask, task, or WBS item. Managers will require control room type information

reflecting cost and schedule status for each discrete task from a computer-based master schedule.

A master schedule must be developed initially to represent the project activities, their flowtimes, and their interdependencies and costs. When the starting date is established, the critical path(s), slack times, and completion dates can be computed. The system must accept status and cost reports from the users to maintain the master schedule.

Responsible managers in all project design disciplines must have the capability to evaluate and report progress on their activities and must be informed of exceptions to the existing master schedules when their schedules are impacted. The system will have the capability to limit job execution based on subtask schedule.

Each discipline in the project design community, beginning with preliminary design, either develops data for its own or for other disciplines' use or depends on data from other disciplines. It may also be involved with a combination of these situations. The master schedule that has been developed is assumed to prevail unless an exception has been posted (e.g., a missed milestone, a report that an activity will be completed, or other notification that information will be available late). In case of an exception, the system will analyze the impact of the exception and print out a report for distribution to each affected organization. It is then the responsibility of each organization to develop a workaround or to confirm the impact. Any changes to the master schedule will be reported to the system.

The interactions with the master schedule will be extended to manufacturing users of IPAD having the same capabilities that are available to the design users.

Manufacturing data is provided to project designers on a committed basis, just as design information is provided to manufacturing organizations. This begins in preliminary design and extends to formal engineering releases and to the support of design changes during production and test phases. The manufacturing user will get the same systems support as do the designers.

USER INTERFACE

The IPAD user interface involves the following basic elements: people (users), tools (application programs, utilities and the system executive), product (information), reports (communication).

IPAD Users

There are seven categories of IPAD user: companies, design and manufacturing engineers, technical managers, application programmers, data base administrators, system administrators, and secretaries. Each of these categories has its separate concerns that must be addressed if IPAD is to be accepted as a product design tool. In addition, there are some general needs that affect all users.

The IPAD system will permit a smooth transition from the existing product design environment into the IPAD environment, permit individual company flexibility in selecting priorities for incremental implementation wherever possible, and provide parameters that can be varied to tune the system to suit individual company needs.

Individual users need a way to store information and have it readily accessible and presentable in either report format or through use of graphic utilities. The user needs to be able to transform known information (input) into desired information (output), either by editing (human action) or by use of a computer program or a linked chain of computer programs. The user should not need to change the input in order to make it acceptable to the computer program; computer programs and data must be integrated. The data owner needs to have confidence that while he is creating his data it will be free from interference by other users unless such interference is knowingly permitted by the owner. When the data is of acceptable quality, the owner may make it available for all users to read through a release procedure. In addition to having design application programs, a user needs some special general application programs or utilities, such as an editor. IPAD will provide a learning capability supported by tutorial texts, programmed learning utility, and example problems. In addition, the system will provide "real time" assistance in using IPAD and will address variations in user skill level, psychological factors involving user confidence, and minimization of user frustration in the use of the system.

Through company-wide involvement, users with all kinds of skills and levels of computing expertise will be involved in the use of IPAD. The system must be able to recognize the user's skill level by the manner of his responses to the system requests and will supply him with appropriate IPAD diagnostics, prompts, defaults, and abbreviations.

IPAD requires little or no user awareness of computer hardware. The user interface will appear essentially the same no matter which computer is being used.

IPAD monitors all instances of data and computer program access and provides a history of all such transactions, recording

the data, who accessed it, and the time of access, thus enabling the system to inform the appropriate user when data changes occur.

Doing Work: User requirements relative to solving work problems are discussed below. A convenient breakdown of these requirements, describing the work process as "prepare," "solve," and "report," has been selected.

Preparing To Solve Problems: The design process can be a useful tool in preparing to solve engineering problems. The activities describing the design process can be linked to a work breakdown structure on the one hand and to pertinent supporting information on the other. Such supporting information will include selection and display of the appropriate data definitions and computer programs. IPAD will provide the user with assistance in creating new computer programs or data in IPAD.

Debugging needs for online programming will be available. These include the ability to (1) determine the value of any program variable, (2) change program parameters while the system is running, (3) determine the status of the data base, (4) change the data base as the system runs, (5) stop the action and then proceed, and (6) execute stepwise.

The system provides a way for the user to assemble data and programs into a desired arrangement which a subsequent IPAD activity might require. Features will be provided that perform or at least facilitate linking computer programs with each other and that enable a user to check the consistency of data and the compatibility of data flow among programs in sequence. The system will permit a user to create and modify definitions applicable to data in the data base.

On request, IPAD will provide the user with the classification for accessing and manipulation of data assigned to him.

Solving Work Problems: The system provides a means of assisting a user in running a job, which, in this context, may be part of a computational task and may involve more than one computer program linked in sequence. The following are examples of activities that will be available while running a job: reviewing intermediate results, overriding pre-established sequences, reviewing final results, selecting final results, and comparing the results of two or more programs or data sets and inform the user of differences with meaningful diagnostic messages.

A job may be run in an interactive mode or submitted for batch processing, or it may be interrupted in the interactive mode and submitted for batch processing at the point of interruption.

Once a job is executing (online or batch), it will be possible to monitor progress as well as to view the output in progress.

Reporting: Reporting refers to the set of activities that assist the user to make existing data visible to members of the IPAD community or to system users. Reporting information is an essential part of the product design communication process. Some of the major needs to be addressed are:

1. A general reporting capability for use in documentation, regular reporting, and real-time displays

2. An English-like display capability for use by nonprogrammers to form visual displays for online reports

3. Timely transmittal of designs created at an iteractive console to more precise computerized drafting machines

4. Conversion of numerical data into graphic format such as multiple x-y plots, bar charts, contour plots, and carpet plots

5. Aids for project management including displays of task assignments, progress, resources expended and available, and milestone schedules

6. Automatic reporting of such things as data due in the data base to meet a schedule date and data that is overdue in the data base

On request IPAD will provide the user with a report on the current status of his computer resource account.

IPAD will also enable the creator or modifier of data, when storing such data, to qualify it for the benefit of other users. The appropriate qualifying label will appear on each page of data printed or displayed. The number and character of the qualifier will be optional in order to satisfy individual company requirements.

Exit IPAD: After completion of a work session and prior to discontinuing computer contact, use of the command EXIT will cause the system to perform the following activities:

1. A report on the status of the data or programs affected by the current work session will be available (printing optional).

2. If the status report shows a need for action on the user's part prior to the end of the session, the system will indicate a command mode situation following the status report.

3. If the user's message file contains any messages, a suitable reminder (e.g., MESSAGE ON FILE) will be displayed.

4. A report on resources used during the session will be available but optional.

Special Human Requirements

This section discusses requirements exceeding the normal user needs as previously described. This includes needs of special interest users interfacing uniquely with the IPAD system as well as special needs of the normal user for reinforcing his system support.

Data Base Administration: The IPAD system must accommodate the special requirements of data administration. The data base administrator is a person or organization responsible for exercising control over IPAD data. This control applies to content, access, storage, definitions, inactive status, integrity, etc. The system must recognize the authority of the data base administrator(s) to access the system and must allow him to perform his functions.

Both system and manual procedures must be developed for the data base administrator, who must be able to access the IPAD system even more quickly and easily than the normal user. The system must recognize his authority and allow him to perform tasks that other users cannot. The written procedures must fully describe the functions and authority of the data base administrator(s).

Secretarial: IPAD will provide a complete capability to enter, maintain, and print documents.

System and Personal Messages: The IPAD system will support both system and personal messages from one user to another. System messages will be displayed automatically at the time of access, and messages concerning impairment of data will be displayed at the time the impairment is discovered. Personal

messages will be retained in the user's personal message file, which may be accessed at any time in the command mode. The user is warned by an increasingly annoying warning that messages are in his personal file.

Conference Viewing of Data: IPAD will provide for multiple or conference viewing of data. To this end, concurrent display of data at more than one terminal will be available. In addition, members of the conference can selectively edit the data being viewed.

User Tools (Utilities)

A number of utility modules will be available to perform services for IPAD users. The utility programs are part of the IPAD system and will be accessible via the IPAD command language. At least the following set of utilities will be available:

Executive and display language processors
Geometry generating utility
Graphic aids
Tutorial aids
Text editing
Menu builders
Project management aids
Report generating
Message processor
Data transfer aids
Usage statistics
Software maintenance
Accounting programs
Arithmetic and logical operations
Computer-aided learning
Program development
Interfacing and integrating computer programs

CONCLUDING REMARKS

This brief description of the engineer's views of IPAD started with a discussion of creating design information using "job" as a building block. "Job" is not an IPAD creation; it is how design work is done with or without computers. To fully appreciate the advantages of implementing "job" in an IPAD environment, one needs a more structured view of how engineers do their work.

The four major IPAD components were described with brief descriptions of the design process and the need for an information

processor to support that process. Design project management was described in some detail; because IPAD is computer-based, it is necessary to interface the user with that system. Figure 7 depicts the four major components of the IPAD system as they relate to an individual job execution.

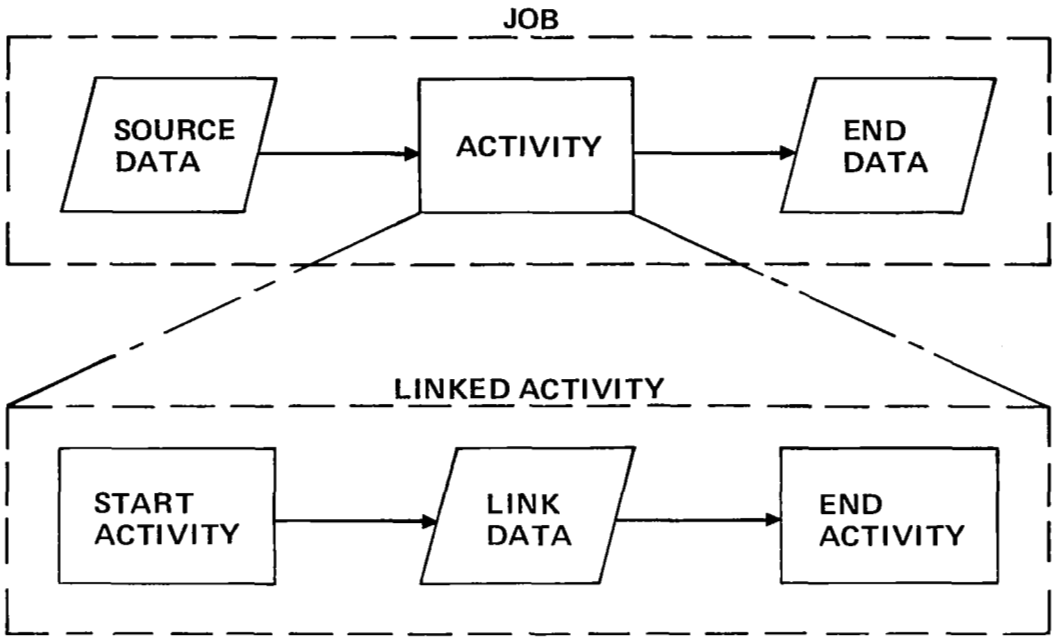


Figure 1.- Job definition.

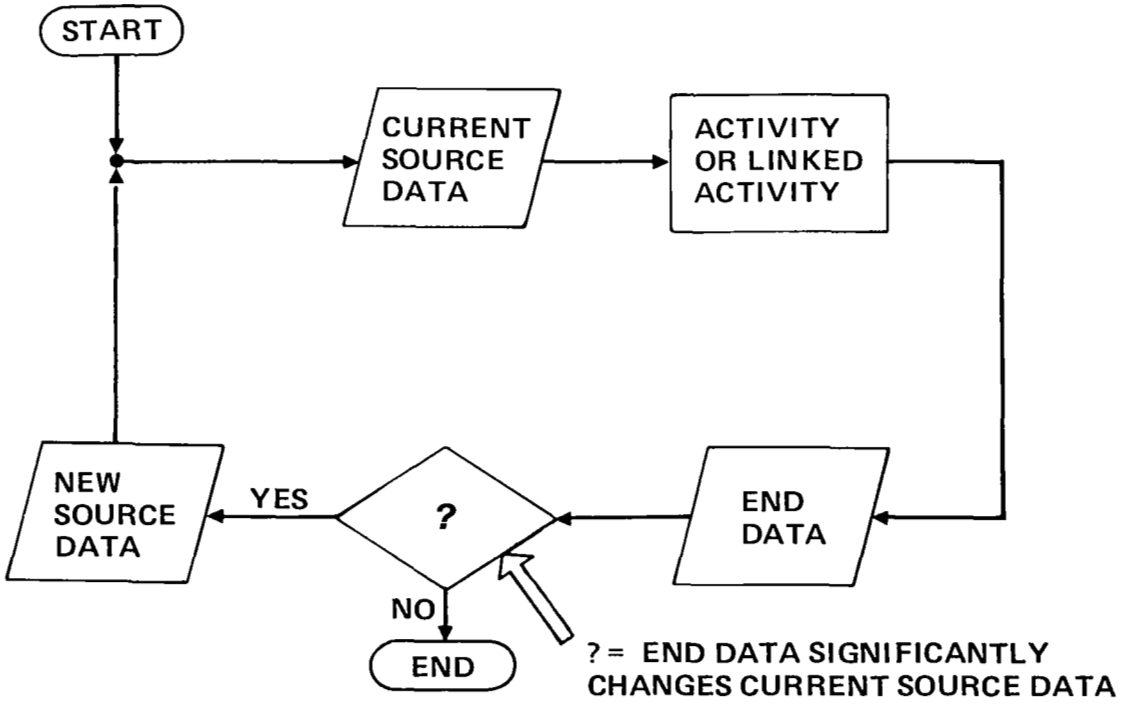


Figure 2.- Engineering data characteristics.

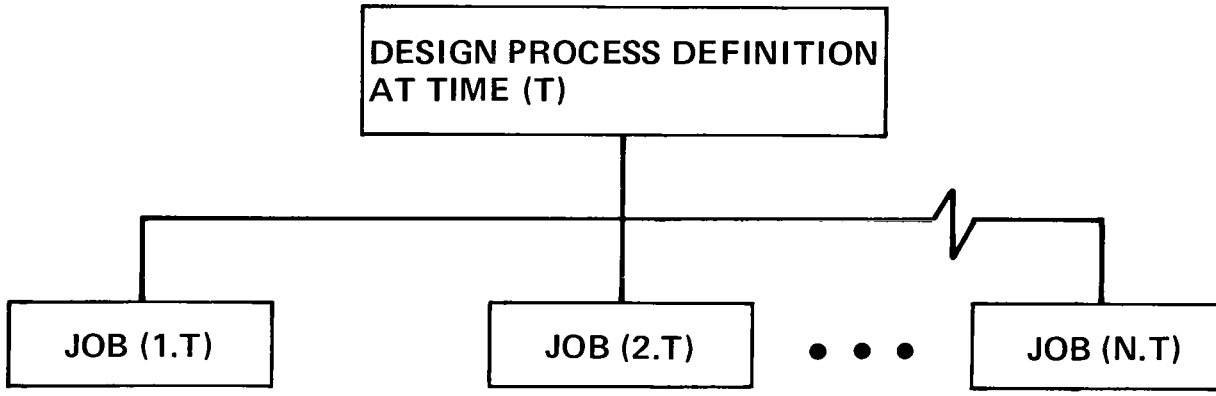


Figure 3.- Design process time slice.

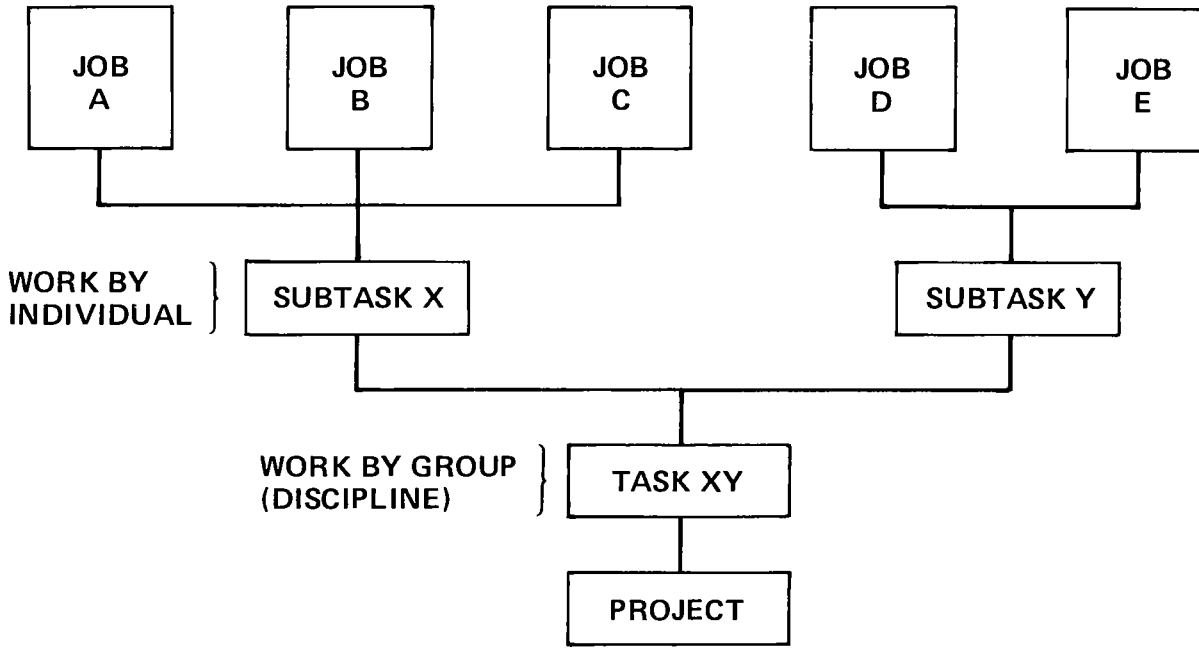


Figure 4.- Job/people relationship.

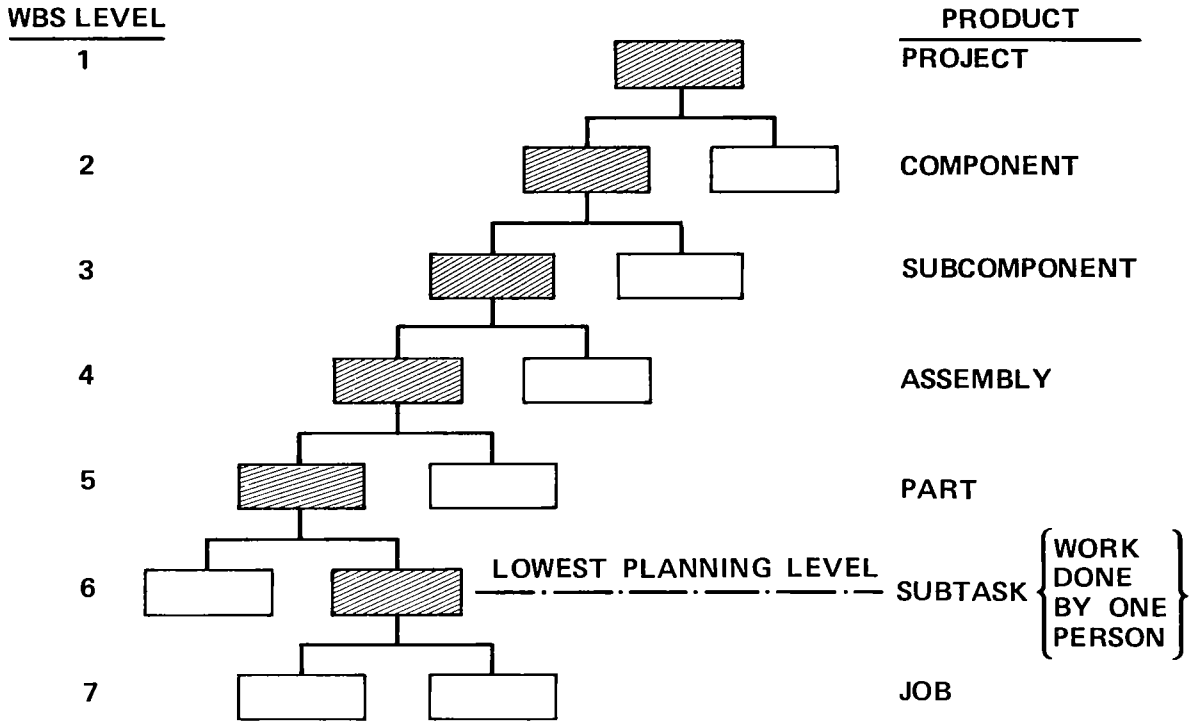


Figure 5.- Job/product relationship.

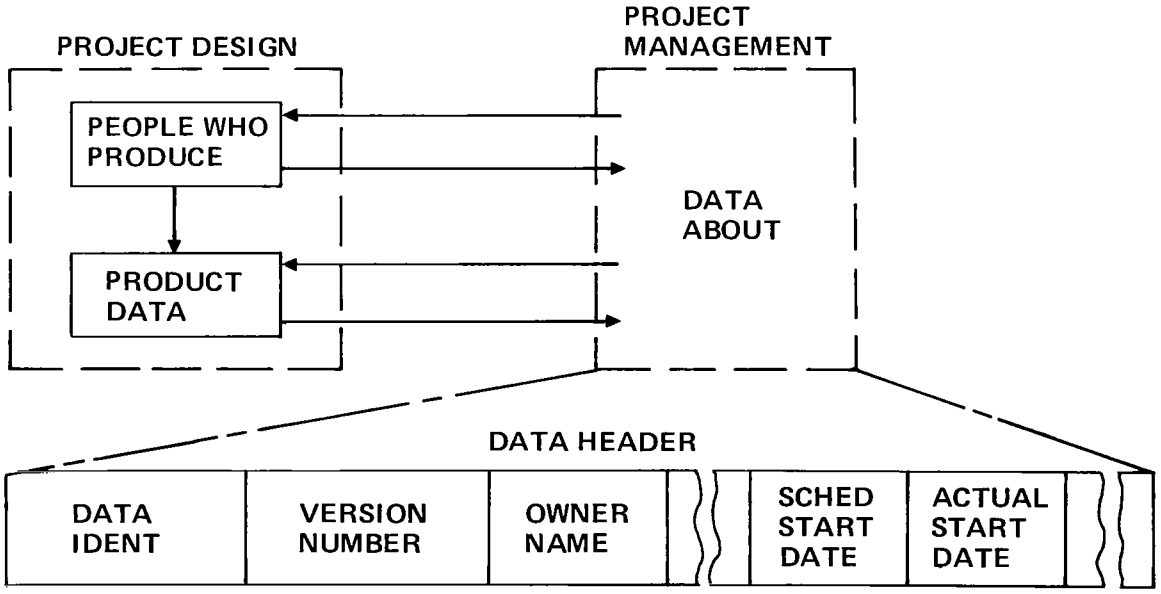


Figure 6.- Project management information.

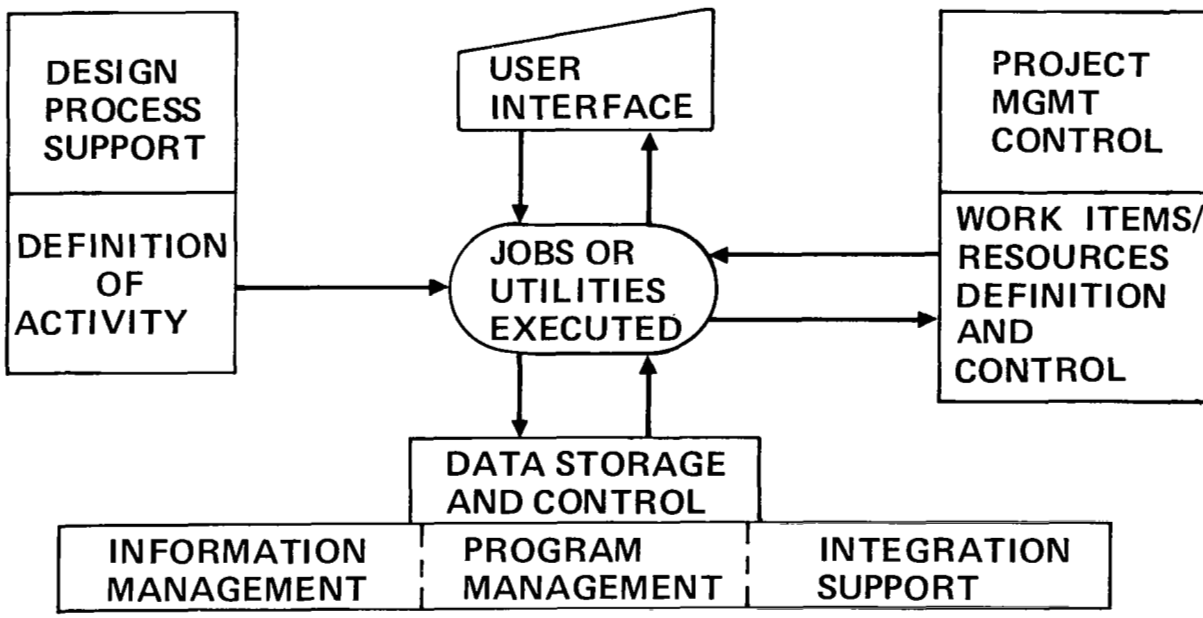


Figure 7.- Functional integration.

FUTURE INTEGRATED DESIGN PROCESS

Donald D. Meyer
Boeing Commercial Airplane Company

SUMMARY

The design process is one of the sources used to produce requirements for a computer system to integrate and manage product design data, program management information, and technical computation and engineering data management activities of the aerospace design process. Design activities were grouped chronologically and explored for activity type, activity interface, data quantity, and data flow. The work was based on analysis of the design process of several typical aerospace products, including both conventional and supersonic airplanes and a hydrofoil design. Activities examined included research, preliminary design, detail design, manufacturing interface, product verification, and product support. The design process was then described in an IPAD environment--the future.

INTRODUCTION

A modern aerospace vehicle is a complex integration of sophisticated technical systems precisely designed and manufactured for safety, economy, and mission performance. The complexity of these vehicles, their processes, and their design and manufacture has increased steadily over time. The design and manufacture of more advanced vehicles are limited by the technology and methods available to develop and manage these processes. Traditional design distribution and automation constrain the technology that can be designed into advanced vehicles. Further, the productivity limitations of traditional design methods increase the manpower required to produce and manage the enormous volumes of necessary information during a reasonable flow time.

Computer-based technology applications have brought about significant improvement in engineering productivity. However, these improvements, by focusing largely on isolated elements of the vehicle design and manufacturing process, have only partially exploited the potential for computational efficiency and automated data communication. There remains the need to provide for expansion and interface to all aspects of design, analysis, testing, in-service monitoring, and manufacturing in order to lower aerospace vehicle development cost, shorten development flow time, and produce a competitive vehicle.

During the late 1960's the use of computers for integration of design and its interfaces evolved as a basic concept. In 1973, a NASA-funded feasibility study of Integrated Programs for Aerospace-Vehicle Design (IPAD) indicated that individual productivity increases through automation and computer support of routine information handling are feasible. Such automation will decrease cost and flow time in the design and production processes and improve the competitive position of the U.S. industry. Consequently, the aerospace and computer industries became deeply involved during the IPAD feasibility studies and continue to guide and critique the program development.

The IPAD objective is to develop a computer-based technology for integration of the design process into a forceful engineering tool with interfaces to manufacturing computerized methods in order to favorably influence aerospace-vehicle performance, development cost, and flow time. To understand this objective, one must understand the design process.

This paper describes the objectives, results, and efforts involved in documenting the design process (ref. 1), manufacturing interfaces with design, (ref. 2), design information processing (ref. 3), needs of the designer (ref. 4), and--looking at the future--integrated design.

The work was done as part of the IPAD program currently under way at the Boeing Commercial Airplane Company.

THE DESIGN PROCESS DEFINITION

The design process (ref. 1) was documented as part of the IPAD task. IPAD, an acronym for Integrated Programs for Aerospace-Vehicle Design, has as its keywords "integrated" and "design." These terms must be defined to ensure that the work direction is effective and to the point.

Design (noun): A planned scheme or arrangement of things or activities leading to a desired end result. The design process is the procedure involved in creating a design.

Design (verb): To create a design where the task is defined (with its environment, restraints, and freedoms) and the best or accepted solution is developed during the available time span with minimum task compromise.

Integrate (verb): Make whole by bringing all parts together.

To make IPAD an effective tool in the design process, it is necessary to describe the design process as a reference, define

the needs of the designer, then harness the computer to help with the job.

DESIGN PROCESS DESCRIPTION

A total design complex consists of research, preliminary design, and product activities. These can be expanded (for convenience) into nine activity levels:

Continuing research:

Level I Research

Preliminary design:

Level II Design criteria selection

Level III Design sizing

Level IV Design refinement

Level V Design verification

Product:

Level VI Detail design

Level VII Product manufacture

Level VIII Product verification

Level IX Product support

It should be noted that, although nine levels are described, their boundaries are nebulous and, in practice, arbitrarily labeled. Figure 1 is an overview of these design process activities.

Each element in figure 1 is displayed in a symbol. Rectangles indicate activities and diamonds indicate decisions. Arrows, of course, indicate direction of the activity's flow.

Continuing Research

The first activity level, continuing research, is in an arena by itself. Included are research activities of a long-term nature involving a search for new materials, configurations, and procedures applicable to many design disciplines. The results are documented for future use. Today they are stored in a filing

cabinet; tomorrow they will be stored in a computer data base for quick and easy access.

Preliminary Design

The second group, preliminary design activities, is divided into four levels: design criteria selection, design sizing, design refinement, and design verification. This group has a goal and a time schedule, both of which become more realistic as the task evolves. The early iterative nature of these levels narrows to a single concept when the last level, design verification, takes place.

The goal of the first of the preliminary design levels is to determine the design criteria (level II) that will result in a product that meets the greatest need and, therefore, has the greatest sales potential. An IPAD-influenced design complex with a dynamic and well-fed data base will define (or provide data to define) the market environment features of existing vehicles and existing and potential route structures, as well as give visibility to configuration combinations of airframe and engineering parameters, computer performance, and design trade data. Such a data base will also define (or provide data to define) airplane economic analysis for selecting the configuration with the best characteristics and will show relative cost data, expected operating costs, route applications, and forecast fleet economics.

The design sizing activity (level III) attempts to size several potential designs to the criteria established in level II and to select candidate designs for further refinement. These design iterations and analysis computations, while deeper than level II, are all limited in detail and broad in scope but are essential for narrowing the design field. In an IPAD environment, geometry is easily calculated, and once it is established, performance calculations are readily made using parametric data for weight and balance, cruise and low-speed performance, costs, and market suitability. If the configuration warrants, structure sizing takes place using a range of configurations from conventional (tried and true) to innovative. Using the candidate structural arrangements, weights are calculated, the structure is sized for static loads, thermal effects, strength, and fatigue. The configuration is examined for flutter and noise footprints using established parametric data. Studies are summarized for performance, finance, and market suitability, which are compared to the mission; then the configuration is accepted, rejected, or modified. In an IPAD environment, parametric data is applied in all discipline areas: geometry, weights, vehicle performance, structure loads and strength, noise, and costs; and application programs are arranged to give data needed for decision making.

The design refinement activity (level IV) is accomplished by applying more powerful analysis capabilities in previously represented technologies and investigating more areas within a given discipline. This reduces risk quickly by more thoroughly analyzing major design areas or controversial design innovations in specific areas.

The goal of verifying the design (level V) is to investigate all phases of the project--design configuration, cost, market, and finance--to facilitate a go-ahead decision that involves minimum risk. This verification is accomplished by exceptionally thorough and vigorous analysis and testing.

A management decision either to proceed with production or to extend design investigation falls between preliminary and production design activities. This decision is crucial. From this point on the expenditure of effort and money will accelerate; if subsequent configuration changes become necessary, these expenditures will be lost, with consequent schedule and cost penalties and customer dissatisfaction.

Product

In the design progression, the third group of activity levels is product-oriented. Detail design (level VI) is oriented to one product. Manufacturing's target (level VII) is to build that product. The product is verified (level VIII) to comply with the designs and their governing regulations. Product support (level IX) is chartered.

Detail design (level VI), using preliminary design layouts, defines and records every part, assembly, and installation that make up the product.

The drawings and data produced for a part must be in sufficient detail to enable any manufacturer to produce a part that is identical (within specified tolerances) to that built by another manufacturer (i.e., the produced parts must be physically, functionally, and structurally interchangeable within specified tolerances). In today's environment, this involves drawing and dimensioning pictures to define each part, gathering geometry from the loft or other layouts, and producing drawings that interface with or contain the needed information. In an IPAD environment, the needed data is in the data base. The part is developed to match, fit, or clear an approved geometry or part by a specified value. Pictures are constructed from the data base from any desired point of view. True dimensions are available between any points by query. Where today the data is delivered by hard copy (drawings and data), the IPAD system will deliver data to anyone as hard copy and data (or as an authorized link to the data base)

so that hard copy and data can be produced, tools made, and cutting tools directed to produce parts.

Detail design involves three phases: (1) layout, (2) evaluation, and (3) preparation and release of formal drawings and data. A brief description of these phases follows.

1. Layout: The layout phase involves extending the preliminary design layouts by refinement or additional layouts so that they are sufficiently complete for a draftsman, engineering aide, or engineer to produce a complete formal definition of the part or parts desired. This activity involves deeper investigation of the concept, loads imposed, joints, and required part sizing.

2. Evaluation: A second phase involves reviewing and evaluating the design layout for function, cost, and producibility. Evaluation may require the use of a configuration model. When the design represented by the layouts is accepted, it is prudent to order any construction materials that are not readily available.

3. Drawings and Data: The production of formal drawings and data involves producing the part description in such a manner that it is as directly useable by manufacturing as possible. The draftsman must know how the part is to be produced, what type tools are to be used, and what assembly procedures are involved. For example, in today's design environment, the drawing and data for a sheet metal part usually consists of the flat pattern, the bend lines, inner and outer mold lines, lengths of flanges, and flange angles. By contrast, in an IPAD environment, programs stored within the data base would prepare the data to develop and depict the flat pattern of the sheet metal part, needing only such additional inputs as gage, bend radii, and flange lengths. It is conceivable that the program would also indicate when and where flange reliefs might be required when the bend line radius is too small for the flange length. Similar program tools will be available for other design disciplines.

Product manufacture (level VII) is the area occupied primarily by the production activities using detail design data produced in the previous activity level. Computer-aided manufacture is a major entity encompassing five general areas and subareas of data handling: (1) production engineering (involving manufacturing plans, tools, quality, and manufacturing standards); (2) fabrication shops (involving product manufacture, product test, product documentation, and product inventory); (3) status and schedules reporting (involving schedule maintenance, forecasts, inventory control, inventory status, and requirements); (4) material procurement (involving vendor selection, purchased material, receiving and inspection, and inventory maintenance);

and (5) accounting (involving cost accumulation and accounting report generation).

During the manufacturing activities, design engineering is available to assist manufacturing by interpreting designs, modifying designs to ease manufacturing activities, refereeing inspection problems, etc.

Product verification (level VIII) concerns the testing of components and completed products in accordance with design engineering direction. The actual testing is performed by manufacturing personnel, who document test applications and results. In an IPAD environment, the data will be put in the data base and extracted as required for reports, background for design revisions, and future designs (levels I, IV, V, and VI).

Product support (level IX) is involved throughout the design activities and continues until the product is no longer in service.

Service history influences new design. Present service activities trigger design improvements and, if done with the customer in mind, assist in selling current and future products.

GATHER INFORMATION

In documenting the design process it was apparent that it follows an iterative sequence: action-evaluation-decision, action-evaluation-decision, until the decision is made to accept, redirect, or stop the design. This iterative sequence came to light in examining the first activity undertaken to perform a task: the "gather information" activity. Not only is this the first design task, but it occupies from half to three-quarters of the design engineer's effort. This step follows the above described iterative pattern (action-evaluation-decision) to define the task; evaluate the description; and decide to accept, reject, modify, or quit. Figure 2 illustrates several variations of the "gather information" activity. Each element in figure 2 is displayed in a symbol; rectangles indicate activities and diamonds indicate decisions.

The following description of figure 2 is oriented toward airplane structural design; however, the activities and decisions are applicable to any design field. The activities are numbered for identification.

G-1. Identify Task Required

"What is the task?" is the first question to be answered. If the task is not defined, this is a gather information task in itself. To identify a particular task related to design project, the designer must:

1. Read the description of the structural element to be designed

2. Read the design requirements and objectives documentation

3. Read the design criteria document

4. Read applicable design guides

5. Review structural arrangements and layouts developed in earlier tasks

6. Read applicable rules and regulations

7. Read schedule and evaluate manpower requirements

G-2. Specify Information Required

Having identified the task, it then becomes necessary to list what is needed to begin design. Any or all of the following items (and/or others) may be required:

Surface definitions: external and internal contours
Location of adjacent structure and interface
Location of systems provisions and interfaces
Loads and load cases
Structural element and joint concepts
Approved fastener lists and standards
Material allowables and strength data
Process and fabrication specifications
Finish and sealing documents

G-3. Review Data Base

Search data base contents for all information specified above. This information is usually project-dependent as well as general in nature, and it may be found in both group and private files.

G-4. Data Base

This file (information processor) is for retention of both general and specific information of interest to engineering. It

consists in particular of that which is necessary for new product development:

Fixed types of information:

- Unit conversions
- Material allowables
- Process specifications
- Environmental conditions
- Structural properties

General types (less static than fixed types, above):

- Design criteria
- Design guides
- Standard parts

Relatively specific but frequently changed types:

- Configuration loft geometry
- Structural arrangement
- External loads
- Structural concepts
- Systems requirements

G-5. Information Available?

Determine the availability of each piece of information required. If it is available, select it; if not, it must be generated.

G-6. Identify Source

For each information item or category identified as available, the source must be identified.

G-7. Retrieve Information

Search the data base at sources indicated in G-4 above for the specified information.

G-8. Display Information

Display information formatted to provide the best understanding of the task at hand. Combine alphanumeric and graphic displays as required.

G-9. Evaluate Information

Review each information item retrieved from the data base for completeness, accuracy, and appropriateness.

G-10. Information Correct?

For each piece of information evaluated, determine whether it is correct or at least sufficiently accurate for use in the design. Does it have a sound basis for credibility, and does it satisfy the needs of the design problem?

G-11. Save Information in Private (or Subtask) Data Area.

Retain the information selected or generated as part of a local file for future use in development and synthesis of the design.

G-12. Information Complete?

Determine whether all of the information required to develop the design has been collected.

G-13. Start Design

At this point, all of the information is available and the design task can start. This symbol (G-13) is any "do" task in any network requiring such action items as develop, draw, calculate, update, etc.

G-14. Repeat?

Determine whether another information item, similar to that just generated, should be selected or generated exactly in the same manner as the last information item. When it is decided that more information is required using the same methods, the progression is to G-15, where the choice is made.

G-15. Select or Generate?

The choice made here is automatic, since it parallels the method used to secure the last information. If information was selected, the procession is to G-7, "Retrieve Information," and if the method used was generation, then the progression is to G-17, "Method Available?"

The progression continues until all generated and/or selected information has been gathered and accepted; then the task begins (G-13).

THE FUTURE INTEGRATED DESIGN PROCESS

An Overview

The design process of the future will be fundamentally the same as that of today: an iterative process that produces designs progressively closer to the design target until an acceptable compromise between quality, time, and money is reached. The primary differences between today's design process and that of the future will be found in the tools used, the quality of design, the ease of data distribution, and the swiftness of accomplishing the design task.

The IPAD design tools of the future that will contribute most to the enhancement of this process are a well-filled, current, and easily accessed data base; a dynamic and graphic geometry generation, storage, and display capability; and the integration of these tools with computer programs for effective problem solving.

Improved Productivity Through More Efficient Use of Design Time

To be competitive, each new generation of a given product should perform better, weigh less, cost less, last longer, require less maintenance, and be produced more rapidly than its predecessor or its competitors. These are all functions of design.

As design time is used and allocated today, a very large portion (as much as 75 percent) of the designer's time is necessarily devoted to what may be thought of as routine and repetitive activities such as querying the data base and extrapolating existing design detail into new configurations. Into the remaining 25 or so percent of available design time, all of the nonroutine portions of the task--the innovative, unique, original, and sophisticated elements of the creative process--must be compressed. To achieve optimum skill utilization and consequent improvements in quality and productivity, the relative amounts of time devoted to these design process elements (i.e., routine and nonroutine tasks) should be reversed.

In an IPAD environment, the routine design elements will be grouped, integrated, programmed, and accomplished in about one-third of the time now required. This means that the remaining

two-thirds of the designer's time will be saved. In other words, two-thirds of the 75 percent now used for routine design details (or 50 percent of total design time) will be added to the 25 percent that is now normally reserved for unique design functions. By this shift, the designer will at last be free to concentrate on those areas where the big payoffs can be expected--the weight, cost, and maintenance reductions and the durability and performance improvements--in short, on the areas where major productivity gains can be realized. (See figure 3.)

The Visual Advantage

Among the most difficult tasks of the designer--and the most critical from the standpoint of the product producer--are the determination of the precise description of a part and the establishment of its spatial location in relation to other components. The geometry capability of IPAD is, therefore, one of the system's greatest benefits to both the designer and the product-producing organization, for it exposes from every viewpoint the physical shape of an object (the part) as well as its fits, interfaces, and clearances. With IPAD, both the designer and the builder of the future will have rapid access to a complete visual and dimensional description of the part via the geometry data base.

CONCLUDING REMARKS

Future uses of the computer in the design process will allow more in-depth design interations over a given period by providing:

1. A data base with capability for almost instant answers to queries for a greater number of designs

2. Geometric capabilities for rapid spatial configuration with unlimited viewpoints to verify shapes, clearances, and fits

3. Rapid and accurate transfer of data to and from interfacing designs to accommodate interfaces and eliminate interferences

4. Manufacturing data on capabilities, techniques, and costs in the fabrication of designs and the procurement of materials

5. Integration of computer programs for solving intricate problems

6. Visibility of designs at desired scales on hard copy and scopes at any time

7. Manufacturing data that can be fed into numerically controlled machines for producing parts or tools.

This paper has attempted to reveal some potential gains, from a designer's viewpoint, that will be achieved in the future through integration of information with activities (i.e., data with the design process). Achievement of these potentials will depend, to a considerable extent, on the individual and on the direction, assistance, and freedom that is afforded by the system. Designers of the future will benefit from a future integrated design process to the extent that they are able, willing, and free to use it.

REFERENCES

1. NASA Contractor Report 2981, Reference Design Process (Boeing document D6-IPAD-70010-D), March 1979.

2. NASA Contractor Report 2982, Product Maufacture Interactions With the Design Process (Boeing document D6-IPAD-70011-D), March 1979.

3. NASA Contractor Report 2984, Integrated Information Processing Requirements (Boeing document D6-IPAD-70012-D), March 1979.

4. NASA Contractor Report 2985, IPAD User Requirements (Boeing document D6-IPAD-70013-D), March 1979.

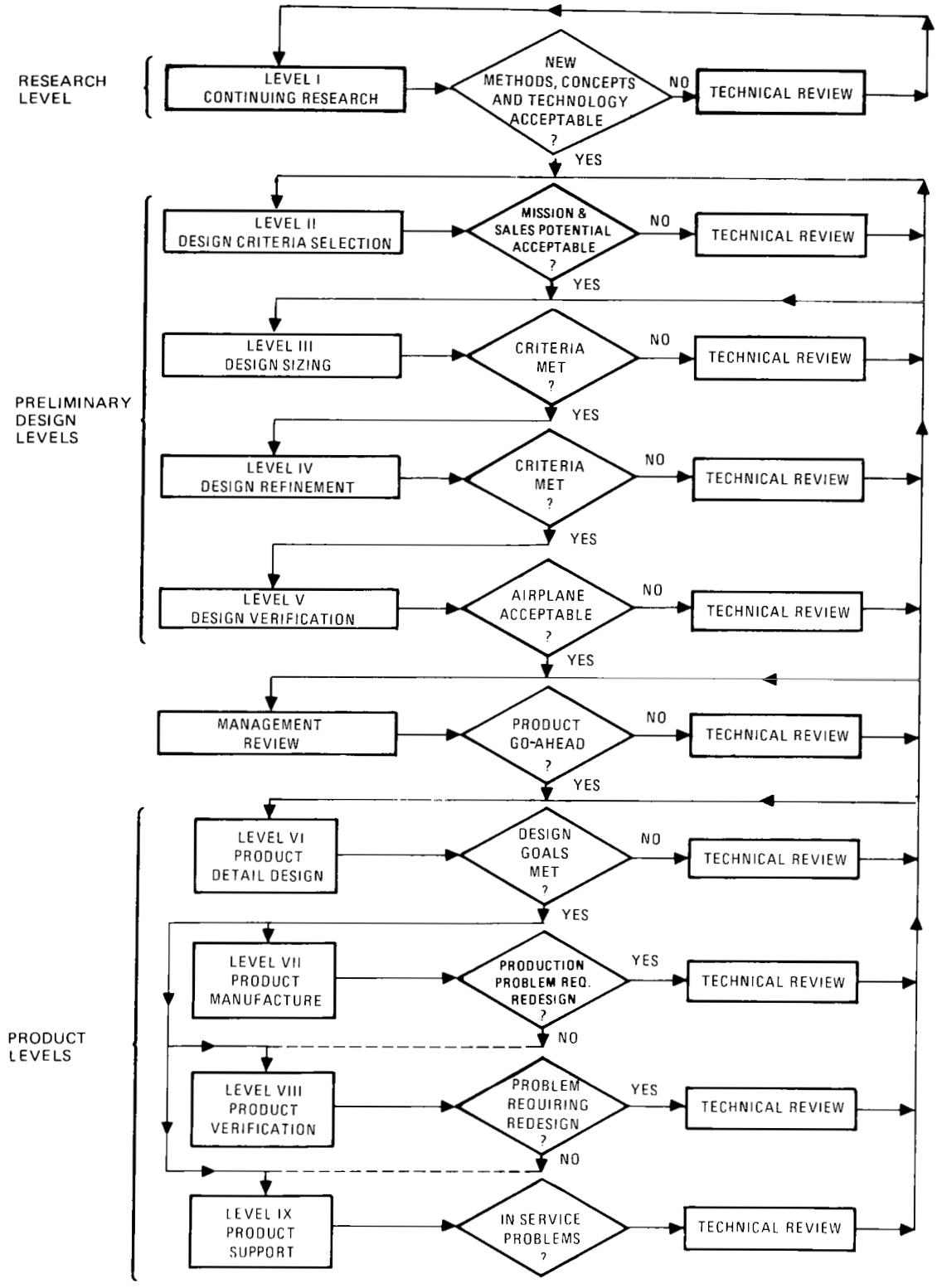


Figure 1.- Activity levels.

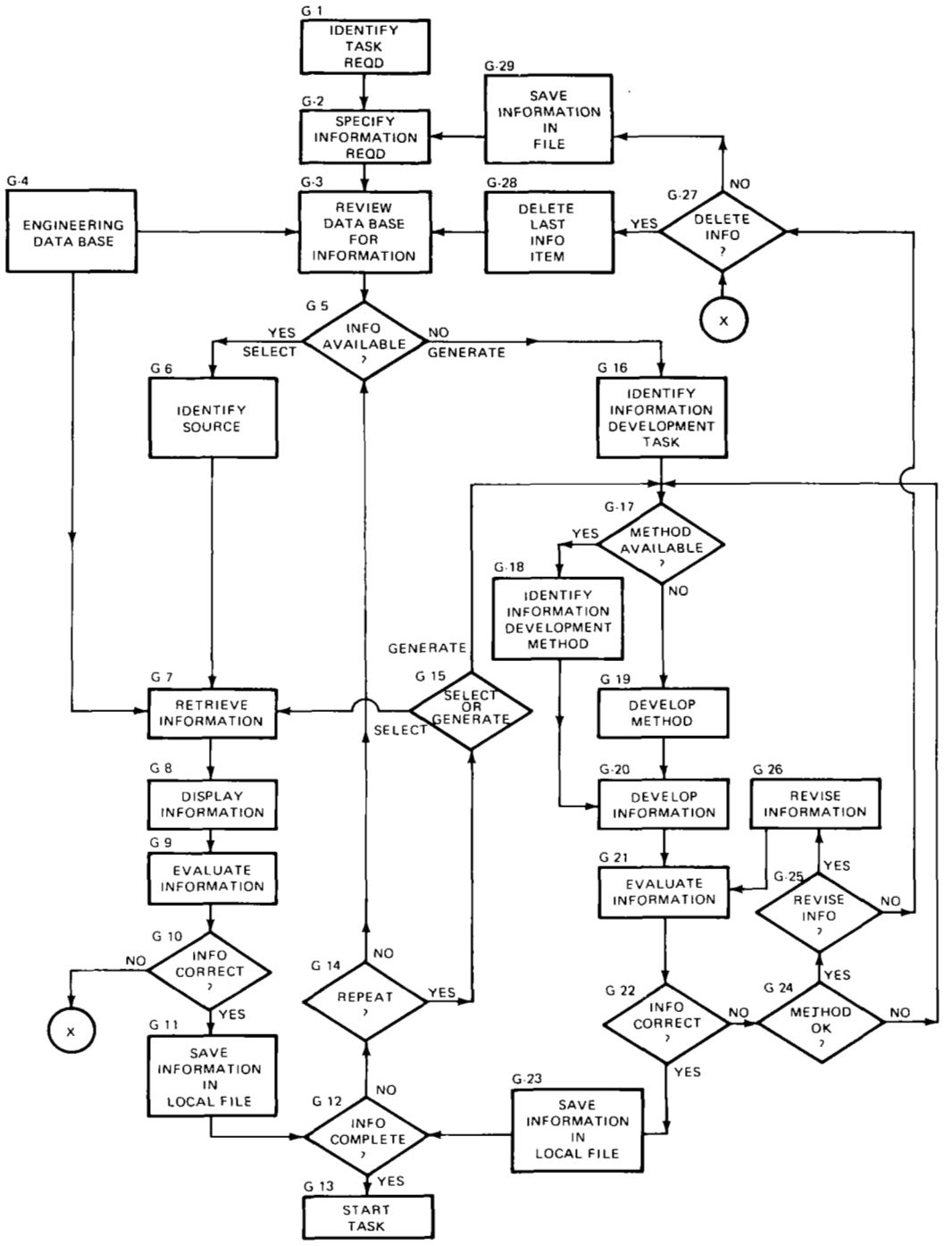


Figure 2.- Gather information network-design.

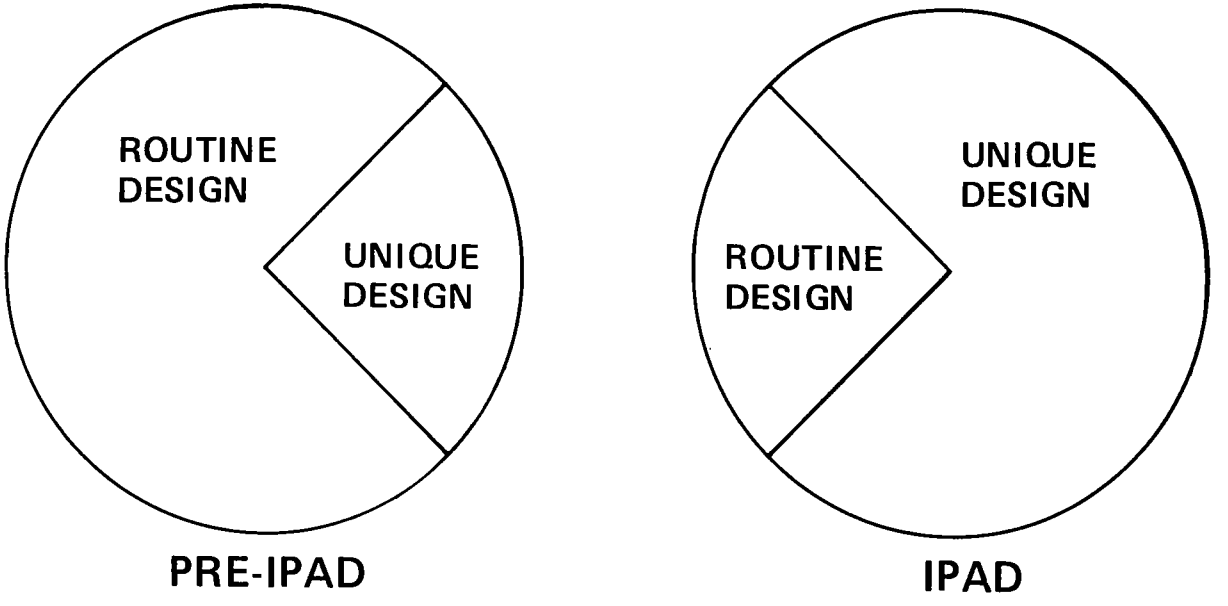


Figure 3.- Design time distribution.

REQUIREMENTS FOR COMPANY-WIDE MANAGEMENT OF ENGINEERING INFORMATION

John W. Southall
Boeing Commercial Airplane Company

SUMMARY

Computing system requirements were developed for company-wide management of information and computer programs in an engineering data processing environment. The requirements are essential to the successful implementation of a computer-based engineering data management system; they exceed the capabilities provided by the commercially available data base management systems. These requirements were derived from a study entitled "The Design Process," which was prepared by design engineers experienced in development of aerospace products.

INTRODUCTION

Engineering data processing has been in a continuing state of evolution that has seen the use of computers expand from problem solving to very large interdependent processing systems. Each new aerospace product line is more complex than its predecessor, and this results in increased specialization and complexity among the engineering disciplines and a greater need for precise communication and comprehensive methodology integration. Computer-based systems have been used successfully to support integration of some design and analysis functions. These systems support data communication between a limited set of related internal functions but are developed with little consideration for management, communication, or control of information outside of the system. Current practice in handling the total engineering data communications relies mainly on human intuition and knowledge resources. However, as the volume of data increases, so do the skills and resources required for maintaining reliability and control of data, and these become a significant cost factor. Thus, the critical items in communication are (1) the volume of information being managed, controlled, transmitted, and interpreted and (2) the effect of the increasing volume of information on cost, response time, and integrity of information.

The solution to the data communication problem will use computerized methods for more effective management of the engineering design information. It is expected that this will have productivity benefits comparable to those achieved by using the arithmetic computational power of the computers in engineering work. While the computational benefits can be realized within isolated subprocesses, the benefits of computerized data

management will materialize only if all data (or at least all major subsets of related data) are managed in a consistent, comprehensive way. Through such integrated management of the engineering processes and their data, the computer will become a more effective engineering design support tool. The IPAD system is being developed to support this capability.

The IPAD requirements for process integration and management of engineering design information are described in this paper. They are derived from analysis of engineering design processes (ref. 1), interaction with manufacturing processes (ref. 2), and management systems used to direct engineering design processes (ref. 3). This analysis produced two distinct sets of requirements. The first set (ref. 4) describes needs to support integration of engineering processes that produce the technical description of a product. These needs fall into four areas: (1) design process support, (2) project management support, (3) information management, and (4) computer programs (software) management. The second set (ref. 5) describes the needs of individual users (engineers, managers, and others) who use computing systems in their daily activities. These needs fall into two principal areas: (1) the user interface with computing systems and (2) the execution of computer programs as user jobs. Figure 1 shows an overview of these requirement areas as they relate to one another. The user interface, project management support, and design process support sections are described in companion papers (refs. 6 and 7). The general characteristics for management of engineering data were presented in a previous paper (ref. 8). This paper describes the IPAD requirements for company-wide management of engineering information in the areas of integration support, information management, and computer program management.

INTEGRATION SUPPORT

The IPAD requirements cover a wide range of diverse user activities. Some examples are (1) simple editing of a text file, (2) creating complex geometric surface models of physical objects, and (3) maintaining an information bank of engineering data (including geometry) in a change-controlled engineering environment. In addition, all of these functional capabilities must be supported in an interactive conversational mode with a common interface language functioning across multiple processors.

A critical aspect of the requirements is the blending of the engineering technical processes and the project management processes into an integrated working environment. This environment must provide semiautomated bookkeeping of project data so that the source, status, and quality of the data developed and used by a design project are identified and readily updated in a

change-controlled data base. This feature must be adequate to relieve the engineer from the routine burden of tracking data. The following subsections describe integration capabilities that must be supported.

User Access

The system must support interactive terminals and provide multiple-user access to all system capabilities through an engineering-oriented language. Tutorial aids must be available to answer user questions. The user interface must identify each user and maintain a user profile data base that provides the primary security control function in the system. User interface commands must also be available to users in the batch mode for such activities as inputting large quantities of bulk data from external sources or executing complex analyses that require extensive computations.

User Control

The capability is required in both interactive and batch modes to execute utility and application programs using procedures initiated, controlled, and terminated from the user interface language. In addition, a high-level terminal session management function is required to provide the engineer with continuity in day-to-day work. It is necessary to allow the user to build up and modify the contents of a private data base over many terminal sessions. These private data bases, identified as subtask data areas, are logically part of an overall information bank but remain private until the user takes action to release the data to a released data area. Released data must be accessible to all approved users, depending on the security assigned to the data. Control functions such as the following are required to support the subtask user:

Change Subtask.-Allows the user to switch to a new subtask from the current subtask without exiting IPAD and logging on again under the new project/task/subtask.

Pause.-Allows the user to halt an IPAD procedure at the next job step. This gives the user the ability to check results at any given point or even step away from the terminal for short interruptions (15 to 20 minutes) and begin again at a logical point.

Suspend.-Allows the user to halt an IPAD procedure, place the activity and corresponding data into storage for periods of 20

minutes to 24 hours, and then resume the application at the point of suspension.

Resume.-Allows the user to return to a terminal session at the point where the subtask was interrupted by a pause or suspend command.

Stop.-Allows the user to halt the session, analyze intermediate results, and back up to a previous job step when it is obvious that the application is beginning to break down.

Quit.-Allows the user to abort the session. This terminates and cleans out all work accomplished with no recovery options.

Process Integration

Capabilities are required for integration of user systems into the IPAD framework. These capabilities will support planning, development, documentation, and maintenance of the processes used by engineering. It is important to be able to describe the interfaces between various engineering disciplines and between the engineering department and other organizations such as finance, marketing, and manufacturing. This relational description must be supported in sufficient detail to identify the data exchange required during the discrete phases of the development cycle of a product. The description must also identify the preferred and optional computer tools available to perform each activity. The data interface description and the input/output required for the computer tools form the basis by which the required data bases can be designed and developed. (See data definition and program integration below.)

Communications

Communications are required for the operation of engineering computing facilities so that data and processing capabilities are (1) readily available throughout the geographic locations occupied by engineering organizations, (2) responsive to user needs, (3) cost effective, and (4) compliant with security requirements. Data (online and archived) and processing capabilities must be adaptive and allow migration of both hardware and software as the computing technology evolves new capabilities.

Engineering Support Functions

Capabilities are required to display engineering data (including geometry) and to support engineers in their creative day-to-day work associated with the engineering design process. The product geometry description provides a common reference for all engineering disciplines involved in the design process. Therefore, a key feature of IPAD will be the capability to describe physical objects in three-dimensional space and to transmit the description of such objects between the various engineering disciplines as well as between engineering and manufacturing processes. To accomplish this, IPAD must provide a standard geometry format and a set of standard utility programs which are state-of-the-art capabilities in such areas as graphics, design drafting, and finite-element modeling. These utilities will be supported by the IPAD system in a manner that provides a unified CAD/CAM capability in which a design may be created, analyzed, and released to the applicable manufacturing process. The standard CAD/CAM utility will enable the user to construct, modify, display, and manipulate geometric definitions. These geometry definitions must be in a form suitable for manufacturing to develop tool path definitions. Figure 2 illustrates some hardware components that are typical of the types required for a computer-aided design work station. It will be possible to display menus on the graphics terminal or on a slave text terminal, as illustrated, and to implement menu selection with function buttons, light pens, or data tablet and menu overlay. The system will access geometry definition for both cut and surface extractions needed for detail parts. Retrieval will be accomplished rapidly in an interactive mode using a language comfortable to the user, such as "DISPLAY REAR VIEW AT BODY STATION 960."

The integration support described in this section must provide a working environment that shifts the computing aspects of engineering design from programmer-oriented languages to engineering-oriented languages. The burden of tracking data and programs must also be shifted from the engineer to the computing support systems. These factors, together with ready access to programs and up-to-date information, will increase the productivity of engineers and improve the quality of solutions. In addition, the engineer will be able to respond to required changes resulting from the normal iterative nature of design by using data change-control features. The following sections present an overview of the requirements for management of information and computer programs that form the basis for development of the integration support presented in this section.

INFORMATION MANAGEMENT

Capabilities are required to administer information as a company-wide resource and to make it possible to blend the engineering and project control processes into a common working environment. An information bank must support schedules, manpower assignments, resource assignments, and critical path definitions as well as provide the repository for design, analysis, and geometry data associated with a project. IPAD must assist engineers in identifying and retrieving information. The engineer will have the ability to request data by name (e.g., wing aspect ratio, engine bypass ratio, cruise Mach number, part number) or to browse through the contents of an information bank by specific disciplines such as configuration design, wing design, loads, stress, hydraulic systems. The following subsections describe information management capabilities that must be supported.

Data Storage and Control

Two modes of data storage and retrieval are required. The first is essentially a file management mode in which IPAD stores and retrieves blocks of data but generally cannot manipulate the contents of the data block. The second is the element management mode, in which IPAD stores and retrieves elements and supports further manipulation of the contents of complex elements, which are arrays with individually accessible items. (See data definition below.)

Data Set

All engineering technical data will be collected by users in units of data called data sets which will be identified and used for change control. Two types of data sets must be supported by IPAD: defined and undefined. Both types require the maintenance of security, version control, source information, quality, and release status in header-like records. The header for each data set includes the unique identification of the data set. Such identification may consist of a descriptor (e.g., finite-element model input), qualifying name (e.g., 727-200 wing box), and version number (e.g., number 4). Thus, the complete data set identification would be "finite element model input, 727-200 wing box, version 4." An undefined data set is known to IPAD only by its header data, i.e., its contents are not defined and it is treated by IPAD as a file. A defined data set is known to IPAD both by its header and its data definition.

Data Definition

A dictionary-type data definition capability is required to support definition of data elements and relationships between elements of defined data sets. The definition for each data element must identify the name, any synonyms, units, and an optional abstract describing the meaning of the data element. The data definition language should be easy to use, and, as a minimum, the following data types must be supported:

Text
Real scalars
Integer scalars
Logical scalars
Dates
Arrays with individually accessible items
Arrays accessible only as complete units
Geometric entities (point, line, curve, and surfaces)

Information Bank

An information bank is required to collect data sets into supersets identified as data areas. The data areas are collected in data bases comprising the information bank. Two types of data areas--subtask and release--must be supported by IPAD. Both types require security and access to the contents of the data areas.

A subtask data area is logically a named private working data base associated with an individual. It is related to an engineering task that, in turn, is related to a design project. The contents of a subtask data area are usually built up by a user over one or more terminal sessions and are logically continuous from initiation to termination. It is essential that the data base management system support the ability to restart a session previously terminated using the subtask control features described above. Subtask data areas are part of the data base; however, they are private and subject to project control until user action is taken to terminate the subtask or release the data generated during the subtask.

A released data area is a group of data sets that have been released and are under version control. Each released data area is named and may be divided into subordinate data areas having a hierarchical structure that may resemble the structure of the engineering organization that owns the released data. The structure is equivalent to a table of contents and can be used in a browsing mode to access data within the information bank. This organization concept is illustrated by figure 3.

Header Data

Header data is required to identify the source and status of data sets and support change control by identifying versions of data sets. Header data contains items such as data set name, version, owner ID, creation date and time, security, retention information, and processing histories. These items allow the user to examine the contents of the data base without listing the data set itself. The contents of header data are defined to IPAD.

Data Release Mechanism

A data release mechanism is required to logically transfer data sets from a subtask data area to a released data area. The mechanism logically resembles the release procedures used by engineering to release drawings, documents, memos, and coordination sheets between organizations and departments within the company. The working security imposed by the release mechanism on released and subtask data sets is illustrated by figure 4. The data release mechanism requires two parts. The first part is the data release process itself. This process involves the act of "signing off" the contents of a data set and may require several levels of signature. The second part of the data release mechanism is the processing actions to be performed on a data set once it is released and includes changing its resident data area's identification and the security locks on the data set.

Security

Security must be provided by user profiles that selectively restrict access to logical portions of the information bank model and by security locks at the data area and data set level. In addition, permission codes associated with users must support selective restriction of IPAD commands.

Backup and Recovery

Integrity must be provided by backup and recovery capabilities of data bases and by logging of transactions sufficient to restore the data base in the event of hardware and software failure.

Data Manipulation

The data manipulation language should be easy for engineers to understand and use. The following four levels of data manipulation are required:

Level 1: Information Bank Operations

Catalog information bank
Combine data areas into logical structures
Modify data area combinations

Level 2: Data Area Operations

Find data area
Catalog data area
Archive data area
Purge data area
Distribute data area

Level 3: Data Set Operations

Find data set
Display data set
Enter/modify data set
Set default values
Audit data set
Copy data set
Share data set
Archive data set
Associate data sets
Purge data set

Level 4: Data Element Operations (defined data set only)

Find data element
Display data element
Enter/modify data element
Copy data element

The information management capabilities described in this section must provide the basis for integration of engineering processes into a common working environment that provides access to a company-wide single-source information bank. The environment must support the multiple views of data required by people and computer programs. It will be possible for data administrators to maintain efficient data storage structures without the need to revise the procedures and computer programs used by engineers.

COMPUTER PROGRAM MANAGEMENT

Capabilities are required to manage the software tools used by engineering. Program management must be similar to the information management described in the previous section. IPAD must provide the engineering user with access to a company-wide application program library. These application programs will be

readily available for execution as jobs that are the tools developed by each company, which have been installed in the IPAD program library. The engineer will have the ability to request job execution by name or to browse through the library by specific discipline and by key words. The following subsections describe software management capabilities that must be supported.

Program Development and Installation

Programs developed within IPAD or suitable existing programs may be installed in the IPAD program library. Application programs conforming to the IPAD installation standard are installed as one or more operational modules. A procedure is used to execute a selected set of operational modules as a job. Naming conventions must result in unique names for all modules and procedures in the program library and must be suitable for use as primary keys for program management. The construction of jobs from modules and the type of administrative information that must be supported by IPAD are illustrated by figure 5. The modular program library must make both source and object modules available to a wide range of users and reduce the need for duplication. Any set of source code in common use should be entered only once as a source language module and should be made available as an object module to the user community. The IPAD program library must provide management of the following module types and execution procedures:

1. A source language module (SLM) will consist of a collection of symbolic source codes meeting the IPAD program library standard for SLM's. Each SLM may contribute to one or more operational modules. An SLM and its corresponding object module must have the same name. These modules must be subject to version control.

2. An operational module (OM) will consist of an executable collection of object modules contributing to one or more user jobs that may be executed under IPAD procedure control. Each OM will be named and subject to version control.

3. An IPAD procedure will provide the capability required to control execution of one or more OM's as a job. A procedure will be named and subject to version control; it will be possible to nest procedures.

Program Integration

Program integration into IPAD will support linking programs to the data within the information bank and to other programs as determined by program use within the design process and the corresponding definition of data flow. This definition will

identify source and destination of all input/output and provide for job-to-job communication. Any required data reformatting or translation identified by this data flow definition will be supported. The source and destination definition will be suitable for mapping storage and retrieval data flow between the process and the information bank and between related activities defined in the process. Both types of data sets will be accessible by programs, as in the case of (1) undefined data sets where IPAD manages data at the level of sets and does not know the contents of the set and (2) defined data sets where data elements within a set and relationships between elements are defined and IPAD manages data at the level of elements. Programs using defined data sets will work on logical views of the data and be indepenent of the physical structure of the data.

Programming Aids

These are required to support creation, maintenance, and integration of application programs and include on-line utilities for program text editing, debugging, and update.

CONCLUDING REMARKS

The computer program management capabilities described in the preceding section will make computing tools readily available for execution by engineers. These tools will be accessible from a company-wide computer program library that supports development of integrated engineering processes.

The integrated computing environment represented by the requirements described in this paper will provide the framework to greatly increasing the productivity of engineers. An engineer will minimize routine efforts of tracking data and programs and will increase time available for creative activities and judgmental decision making. Using a single-source company-wide information bank will enhance a company's ability to transfer data within an engineering discipline, between engineering disciplines and between engineering and other departments of the company.

REFERENCES

1. Meyer, D. D.: Reference Design Process, NASA Contractor Report 2981, 1979.

2. Crowell, H. A.: Product Manufacturing Interactions with the Design Process, NASA Contractor Report 2982, 1979.

3. Isenberg, J. M., and Southall, J. W.: Product Program Management Systems, NASA Contractor Report 2983, 1979.

4. Southall, J. W.: Integrated Information Processing Requirements, NASA Contractor Report 1984, 1979.

5. Anderton, G. L.: IPAD User Requirements, NASA Contractor Report 2985, 1979.

6. Anderton, Gordon L.: Integration of Design Information. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 5 of this compilation.)

7. Meyer, Donald D.: Future Integrated Design Process. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 6 of this compilation.)

8. Miller, R. E., Jr.; Southall, J. W.; and Wahlstrom, S. O.: Requirements for Management of Aerospace Engineering Data. Presented at the Symposium on Future Trends in Computerized Structural Analysis and Synthesis, Washington, D.C., Oct. 30-Nov. 1, 1978.

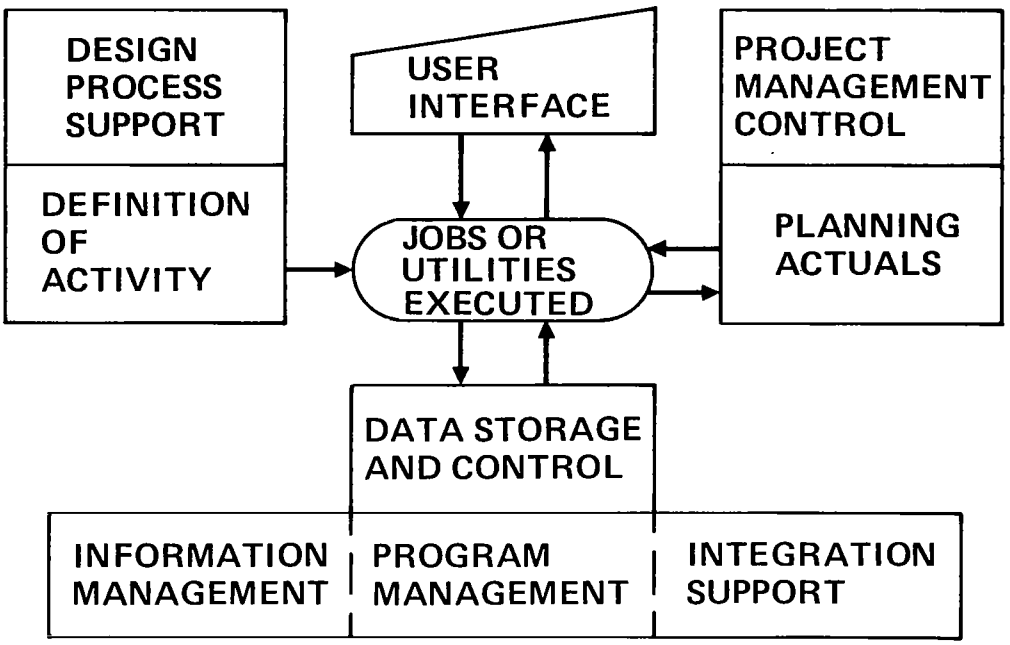


Figure 1.- IPAD integrated environment.

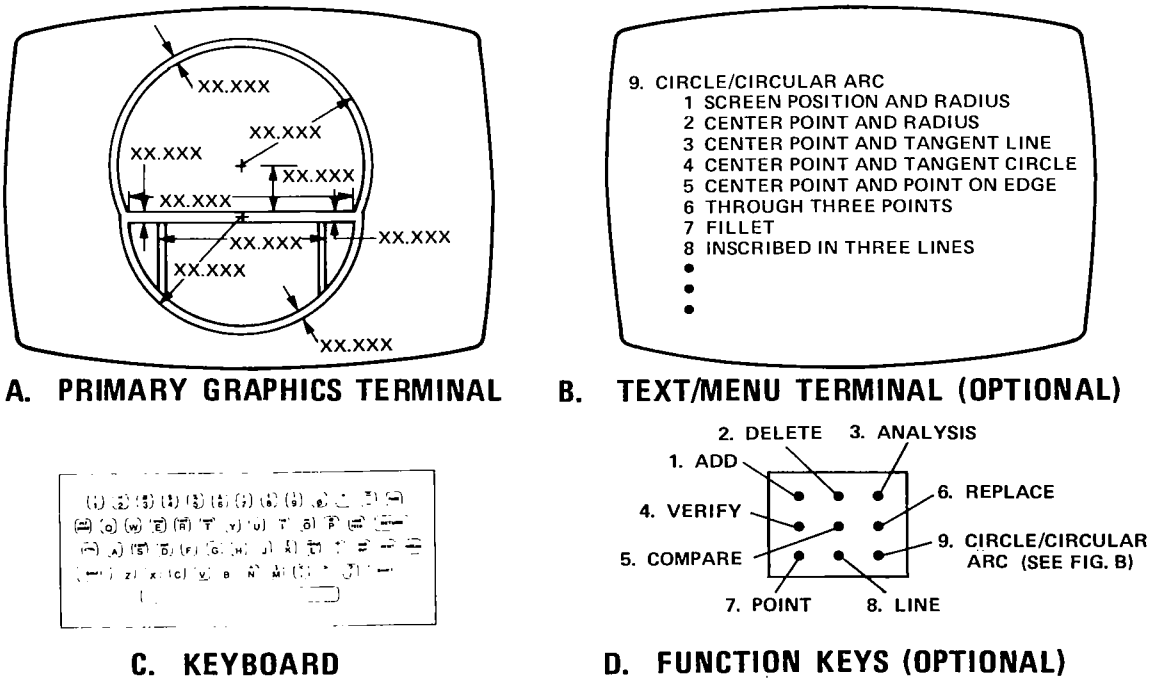
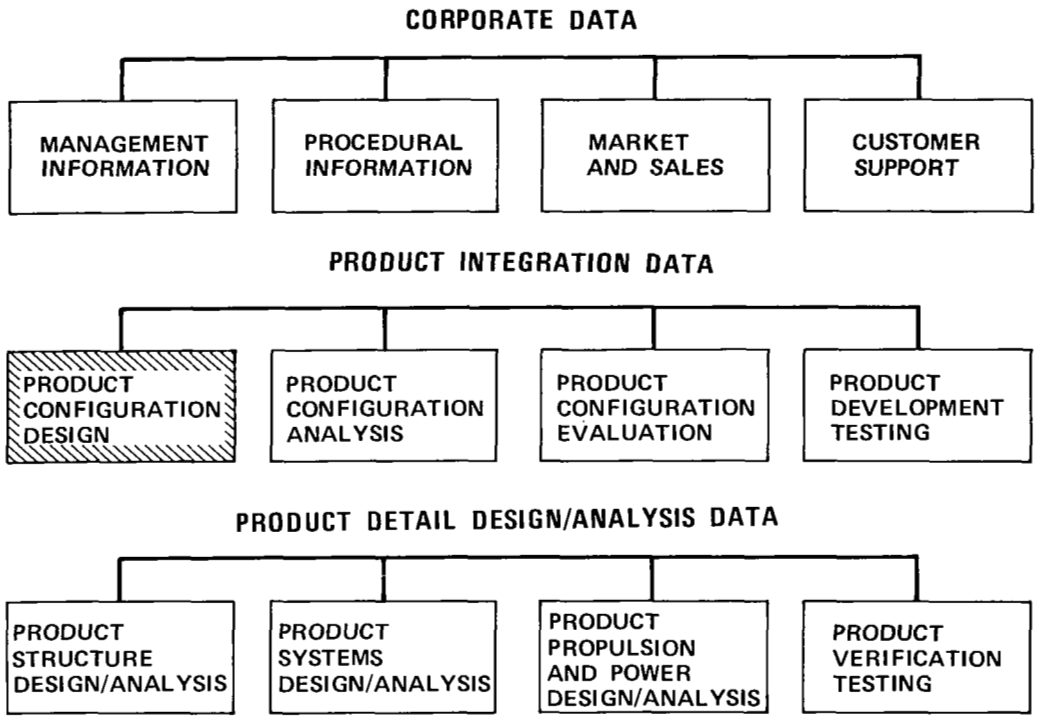
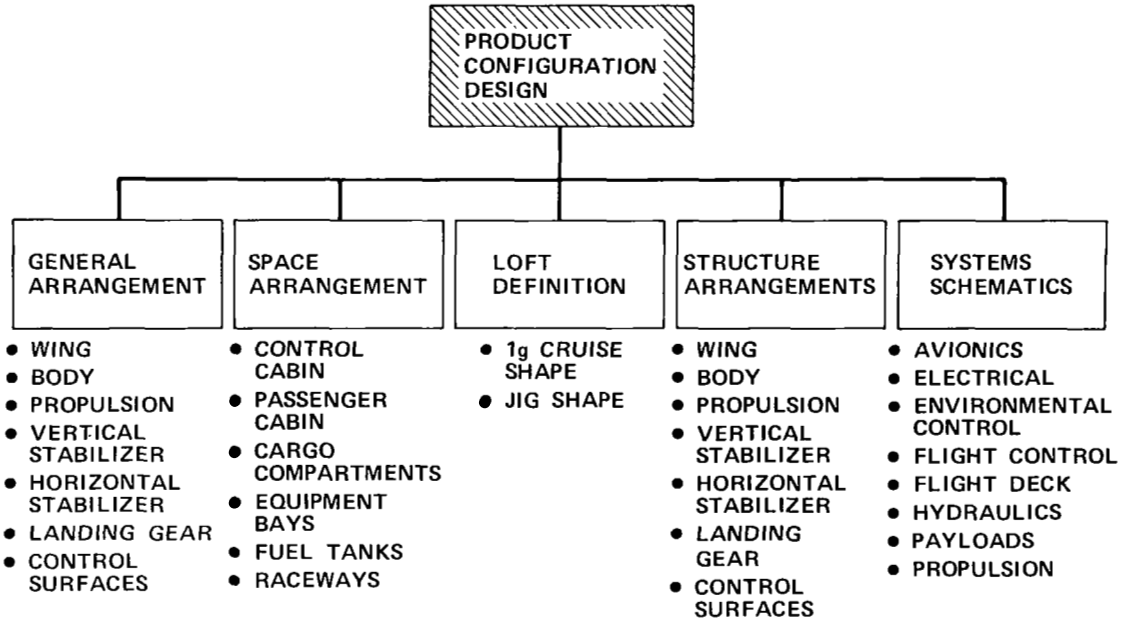


Figure 2.- Typical equipment - CAD work station.



(a) Information bank organization.



(b) Data area organization.

Figure 3.- Example information bank structure.

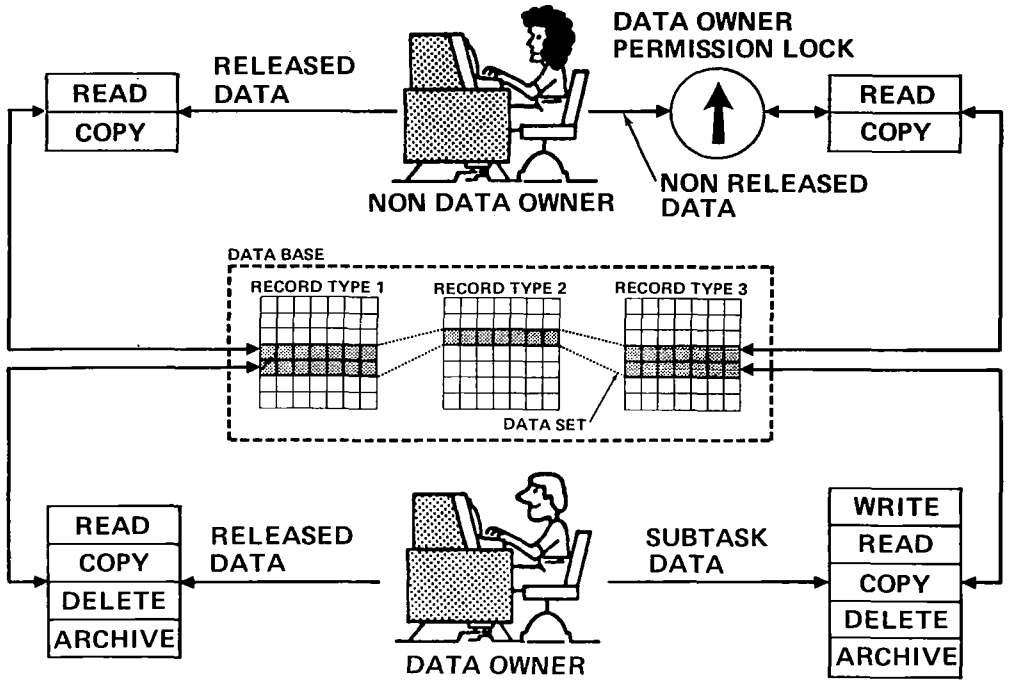


Figure 4.- Working security - released and subtask data sets.

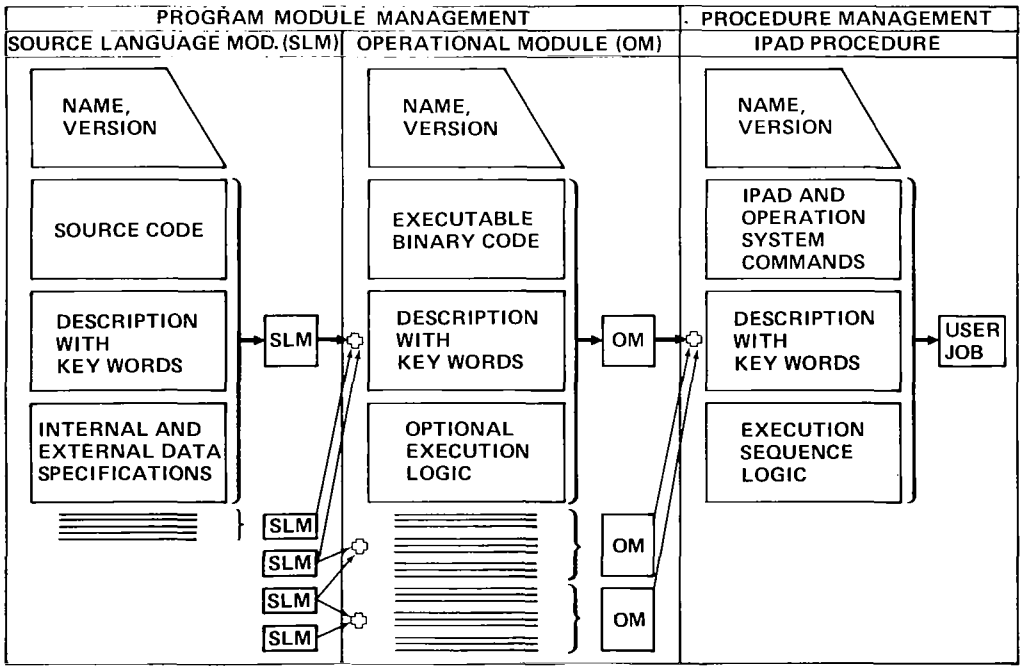


Figure 5.- Software management.

PRELIMINARY DESIGN OF A FUTURE INTEGRATED DESIGN SYSTEM*

Ralph M. Diggins
Boeing Computer Services Company

SUMMARY

IPAD is a system of computer programs and data supporting the aerospace-vehicle design process by providing a set of services to aid in the management of a design project, project technical work, and project support work. Its purpose is to integrate people, programs, and data into a unified aerospace-vehicle design system. All project-management and technical data, together with certain standard data, are stored in a data base. The IPAD functions allow project personnel to query the data base and to perform operations on the data. This permits the orderly sequencing of the task elements of a complex operation and provides common access to a single data base by various participating groups who otherwise would require many separate files. These capabilities will be provided on a single host computer or across multiple heterogeneous computers on a distributed progress basis.

INTRODUCTION

This paper is intended to provide an overview of the preliminary design of full IPAD--a future integrated design system. It presents the level II design, the component functional capability, and a subset of full IPAD, including its hardware and software architecture.

IPAD is being developed by The Boeing Company under contract to NASA Langley Research Center. The five-year contract was awarded in December 1975, and work commenced in April 1976.

The IPAD preliminary design is a direct response to the requirement specifications that were developed by The Boeing Company with the advice and concurrence of ITAB and NASA. The preliminary design, as it exists today, portrays an evolutionary state-of-the-art system structured to support the engineering design process in all its complexity and sophistication as shown in figure 1.

* Summary of design response to the level II IPAD requirements set forth by NASA contract NAS1-14700.

IPAD is the key component of a computer-based engineering complex (CBEC). It controls and provides all of the essential services of the complex and, when combined with a company's application programs and appropriate computers, forms an advanced system to support engineering design (ref. 1).

A company may configure a CBEC using many combinations of components. A CBEC may consist of one or more processing elements (PEs). If more than one PE is used, a local network is required. The PEs may be any computing system containing IPAD together with the required local network interface hardware and software components. The PEs need not operate with the same IPAD software configuration (except for a minimum IPAD), a feature that permits dedication of certain PEs to specific tasks such as data management, graphics functions, user interface functions, application programs, software development, etc.

Two or more CBECs may be connected over distances ranging from a few feet to hundreds of miles and will communicate using an IPAD protocol. Communication between CBECs will be handled at low to medium data transfer rates by the host computing system teleprocessing facilities.

A CBEC has access to a data base under the management of an IPAD data management system. Access to this data base is only possible through this data management system; however, local data bases not under IPAD data management control can be used by the PEs.

IPAD is being developed as a distributed computing system capable of operating in many configurations, including single-processor configurations. The IPAD functional components may be placed on the PEs within the CBEC to best suit the needs of the complex. The local network provides the data transmission medium, when required, between IPAD functions.

IPAD operates under the control of a host computer operating system as a batch system, an interactive system, or both. The primary mode of operation will be interactive, provided the host system supports interactive processing (ref. 2).

Input and output devices will be attached to a host computing system. Actual device input and output are accomplished by the host, while IPAD performs input and output with virtual devices. IPAD resolves any inconsistencies between host device-dependent input/output and IPAD device-independent input/output.

Other programs may operate on the same PE with IPAD. These programs and IPAD may be completely independent, or they may communicate indirectly with each other, allowing applications programs not integrated into IPAD to use certain IPAD facilities.

LEVEL II IPAD SYSTEM DESIGN

At level II the system design is presented independent of any particular host computing system. Later this paper will describe modeling the design to multiple host computing systems (ref. 3).

System Components

IPAD is composed of six sets of major components (shown in figure 2), which are: (1) The IPAD executive (IPEX), (2) a set of system functions, (3) the user interface, (4) the IPAD data management system (IPIP), (5) a set of user functions, and (6) a set of application programs installed by each company.

General Descriptions

IPAD Executive (IPEX).- The IPAD executive provides a standard set of services to the other major components, completely isolating them from the host computing system and providing internal functions and data needed for the distribution of IPAD functions, data, and application programs.

The standard services provided by IPEX are process control, input, output, file operations, interprocess communication, and access to certain host resources and services. These services are provided by IPEX through standard calling sequence to the subprograms in a service library. This interface is the same on each host computer, thus ensuring a high degree of portability of the major components (ref 4).

IPEX isolates the other major tasks from the host computing system and provides standard interfaces to all of its services. It makes the transformation between IPAD pseudo-input and -output devices and the software drivers for real input and output devices. It maintains information on the actual location of all IPAD-controlled data.

IPEX also provides the internal functions and information needed for the distribution of functions and data. The computing systems selected to host IPAD do not support distributed computing; therefore, this capability has been incorporated into IPEX. The major IPAD components, the users, and the host computer operating system are not aware of the configuration of the IPAD distributed computing system; only IPEX needs this information.

IPEX is implemented on every PE (host computer) in a CBEC. The control of the complex is replicated in each copy of IPEX. Timely data is maintained on the configuration of the complex, the

status of that configuration, and the location and status of IPAD-controlled files. This data is maintained and used by IPEX to control the distributed computing system (ref. 5).

System Functions.- The system functions provide internal services to IPAD functions such as the collection of performance data, putting such data in files for later processing, collecting user data, etc.

User Interface.- The user interface handles all of the dialogue between the system and the user except when the user is interacting with a user function of an application program. It is the first component to interact with the user when the user enters IPAD and the last before the user leaves IPAD. It prompts the user, reads and interprets commands, executes executive-level commands, aids the user, maintains a profile of the user, controls access to IPAD, provides usage statistics, handles certain abnormal situations, and terminates the user upon request.

The user interface may be installed on one or more PEs in the computer-based engineering complex. It provides the user a view of IPAD that is the same on every PE.

Data Management System.- The IPAD information processor (IPIP) is divided into two sets of functions: (1) the data definition, data manipulation, query, and precompiler functions; and (2) the data management system. The first set is implemented as user functions. The data management system is implemented as a separate major component. Together, these two sets of functions provide controlled access to IPAD data to users, to IPAD components, and to application programs. The data management system operates on logical files and logical data distribution. It requests operations on files from IPEX, and only IPEX knows the actual location of IPAD data.

User Functions.- The IPAD user functions provide support to the design process. These include functions for project management, design, training, data definition, data manipulation, query, precompilation, application program development, and document preparation. Each function contains its own user interface. Functions are selected by users while in the executive command mode. A function returns control to the user interface upon request from the user. In certain abnormal cases, the system takes control from a user function. User functions may be installed on any of the PEs in a CBEC, and a specific user function may be implemented on several PEs.

Application Programs.- A company may extend the set of standard IPAD user functions by installing its own application programs. An application program must contain its own user interface and is selected by a user while in the executive command mode. An application program returns control to the user interface upon request from the user. In certain abnormal cases, the system takes control from an application program. An application program may be installed on any PE with which it is compatible and may be installed on several PEs.

Access Relationships Between Users, IPAD Tasks, Data, and Host Operating System (OS).- The user interface, user functions, and application programs are directly available to users through user languages. The system functions, IPAD executive, data management system, host operating system, and IPAD data are not directly accessible to users. These relationships are illustrated in figure 3. (The access relationships shown in figure 3 are not to be confused with actual system interfaces described later in this paper.)

COMPONENT FUNCTIONAL CAPABILITY

This section contains the functional design capabilities for each of the major IPAD components. This includes interfaces, output, input, functions, and considerations for the effects of function and data distribution on design.

IPAD Executive (IPEX)

Interfaces.- Major IPAD components with which IPEX interfaces include system functions, user interface, data management system, user functions, and application programs. IPEX also interfaces with the host computer operating system (OS). It is the only IPAD component that has a functional interface with the OS. IPEX receives control from the host OS upon its initiation and returns control to the host OS on normal termination of IPAD operation. It handles all data flowing between the major components as well as data flowing between IPAD and the host OS. An exception is an application program that is only partially integrated into IPAD, which may interface directly with the host OS to obtain certain services and resources.

Output.- Output from IPEX is directed to the host OS and to each of the major IPAD components. These outputs are classified according to their destination.

Output to the host operating system:

Task control information
Requests for service
Output from major components destined for user terminals
Output from major components destined for files
Messages to other major components

Output to major IPAD components:

Status and error codes returned in response to requests for service

Service completion block (SCB) returned in response to a request for service

Data from a file obtained by a service request to IPEX

A message from another IPAD component

Output to other IPEXs:

Messages to IPEX on another computer in the CBEC

Input.- Input to IPEX is received from the host OS and from other major IPAD components. These inputs are classified according to their origin.

Input from the host OS:

I/O completion signals
A terminal connect signal
An abort signal
Task status information
Error codes
Responses to service requests
Data from files
Data from user terminals
Messages from other components

Input from major IPAD components:

Requests for service
Messages to other components
Output to user terminals
Output to files

Input from other IPEXs:

Messages from IPEX on another computer in the CBEC

Function.- The IPAD executive provides other major components with standard services including process control, input, output, file operations, interprocess communication, resource utilization data, and access to certain host resources and services. IPEX also provides the internal functions and data needed for the distribution of IPAD functions, data, and application programs.

Upon receipt of a request for service from a major component IPEX, checks the request and, if it is correct, locates the service and forwards the request to the site of the service via some host computer in the network. It resumes the operations of components that correctly request available asynchronous services and leaves suspended those that correctly request available synchronous services. Requested services include terminal input, terminal output, file operations (open, close, read, write), send a message, receive a message, and initiate a task.

System Function Interfaces

The system functions interface directly only with IPEX. They are initiated, controlled, and normally terminated by IPEX, and they obtain all services and resources from IPEX. All data flows between the system functions, and other major components are handled through IPEX. Through IPEX, the system functions may interface indirectly with all of the major IPAD components. They provide a special set of services to the other components.

User Interface

The user interface communicates directly only with IPEX. It is initiated, controlled, and normally terminated by IPEX, and it obtains all services and resources from IPEX. All data flowing between the user interface and other major components are handled through IPEX (refs. 6, 7, and 8).

The user interface indirectly interfaces with all major components and obtains special services from the system functions. Through IPEX, it initiates the user functions and application programs and obtains data management services from the data management system. Upon normal termination, the user functions and application programs return control to the user interface.

Output.- Output from the user interface is directed to IPEX. This output consists of pseudo user-terminal output, messages to other components, and requests for service.

Input.- Input to the user interface is received from IPEX. It includes pseudo user-terminal input, messages from other

components, status and error codes, and results of service requests.

Function.- The function of the user interface component is to provide an interface between the user and IPAD. It is the first and last component with which the user interacts. The user communicates with IPAD using the IPAD command language. The user interface prompts the user for input, interprets that input, and performs the requested function. The user interface determines the authority of users to access IPAD. It provides aids to the user, maintains a profile for each user, and keeps data about a user session.

The specific functions of the user interface component are:

Prompt user
Read terminal input
Interpret input
Log authorized users onto IPAD
Maintain a user profile
Establish user's terminal environment
Log user off IPAD
Provide assistance to user
Execute commands
Pass control to requested user functions
Pass control to application programs
Collect data on user session
Provide reports to user on resource usage

Data Management System

The data management system interfaces directly only with IPEX. It is initiated, controlled, and normally terminated by IPEX, from which it obtains all services and resources. All data that flows between the data management system and other major components is handled through IPEX.

The data management system interfaces indirectly with, and provides data management services to, all major components.

Output.- Output from the data management system is directed to IPEX and includes requests for service, data to be stored in files, and messages to other components.

Input.- Input to the data management system is received from IPEX and includes data from files, messages from other components, status and error codes, and results of service requests.

Function.- The function of the data management system is to provide access to IPAD data as well as control over such data. There are several modes of access to data via IPIP as well as various levels of control. IPAD data falls into four major categories:

1. Data generated by users or user programs where the data type is known to IPIP (i.e., the data itself is described to IPIP by means of a data definition language)

2. Data generated by users or user programs where the type of data is not known to IPIP (files for which a data description of the contents does not exist)

3. Source and object programs

4. IPIP system data that describes user data

Access and Control.- IPIP allows access to all types of data described above. However, access is restricted to some of the IPIP system data for control purposes. Data categories 1 and 2 may be accessed directly by users via an end user query facility or by application programs. Programs are stored into and retrieved from the IPAD data base via a user query facility. All communications (requests, data) are channeled through IPEX.

The data management system provides control over the data in the IPAD data base. This control falls into three major categories:

1. Security--IPIP imposes restricted access to both user data and the IPIP system data.

2. Configuration control--IPIP possesses special capabilities, enabling it to provide configuration control over the data. (IPIP monitors to determine who modifies what and when.)

3. Backup and recovery--IPIP provides mechanisms to recover from system software or hardware crashes and user errors.

The functions provided by IPIP are available in a centralized or distributed environment.

User Functions

The user functions interface directly only with IPEX. They are initiated, controlled, and normally terminated by IPEX, and they obtain all services and resources from IPEX. All data that flows between the user functions and other major components is handled through IPEX.

The user functions interface indirectly with the system functions, the user interface, and the data management system. They obtain special services from the system functions, are activated by request of the user interface, and, upon normal termination, return control to the user interface. They obtain data management services from the data management system.

Application Programs

The application programs interface directly only with IPEX. They are initiated, controlled, and normally terminated by IPEX; and they obtain all services and resources from IPEX. All data that flows between application programs and other major components are handled through IPEX.

The application programs interface indirectly with the system functions, the user interface, and the data management system. They obtain special services from the system functions. They are activated at the request of the user interface, and, upon a normal termination, they return control to the user interface. They obtain data management services from the data management system. Application programs that are only partially integrated into IPAD may interface directly with the host OS to obtain certain services and resources.

The Effects of Function and Data Distribution on Design

From the foregoing it is apparent that the user has no direct interaction with IPEX, the system functions, or the data management system. IPEX affects the flow of control between the host and the IPAD components and among the IPAD components.

It is important to note that there are no direct interfaces between the major IPAD components (system functions, user interface, data management system, user functions, and application programs). An indirect interface is provided through IPEX. To understand the reasons for this, a review of the IPAD system architecture is necessary.

IPAD was designed as a distributed computing system, as evidenced by the following:

1. There are several interconnected hardware elements.

2. IPAD components may be distributed over processing elements (PEs).

3. IPAD-controlled data may be distributed onto storage devices attached to several PEs.

4. Control of the IPAD system is replicated on each PE.

An installation may distribute functions, application programs, and data to best suit its needs.

Since IPAD components can be distributed throughout the system in various configurations, there can be no direct control or data interfaces between them. These interfaces must be provided indirectly by the operating systems on the PEs. The operating systems for the computing systems selected to support IPAD do not support distributed computing. These functions must be provided by IPAD as an extension to the host OS. IPAD has been designed so that these functions can later be assumed by the host OS with minimal changes to all components except IPEX. Figure 4 shows IPAD on two interconnected PEs.

The IPAD components depicted by A, B, C, D, and E of figure 4 have been loaded onto PE 1, and components F, G, H, I, and J have been loaded onto PE 2. These represent system functions, user interface, data management system, user functions, and application programs. Many other combinations of these components are possible on the PEs. To ensure maximum flexibility of IPAD, an installation must be free to distribute components to best satisfy its requirements; therefore, the components cannot expect to find each other on specific PEs, which makes it impossible to establish direct interfaces between such components.

The software components in a computing system are either elements in a sequence of operations, or they provide services to each other. In nondistributed computing systems, these components are usually linked together to form a contiguous block of instructions and data. The various components (main program and subprograms) are executed through a series of subprogram "calls" and "returns." They pass data among themselves using parameter exchanges by calling sequences, accessing global data areas, or auxiliary storage devices.

IPAD components must interact with each other through IPEX, which alone has access to the information concerning the location of programs and data. This information is kept in the system configuration tables.

Assume that in figure 4 the letters represent the following IPAD components:

LETTER	COMPONENT
A	System functions
B	User interface
C	Text editor

D	Data base query
E	ATLAS
F	System functions
G	Data management system
H	Data definition function
I	FORTRAN precompiler
J	Design drafting function

The system configuration tables reveal the actual location of these components and are accessible by IPEX on each PE.

Obviously, from an internal systems viewpoint, distributed computing produces higher overhead costs than do nondistributing systems. The performance of a properly configured distributing computing system, as perceived by a user, may be far superior to that of a simple system when both have many users and are providing equivalent services.

The need to distribute the major IPAD components over the set of PEs in a distributed computing system also suggests that the major IPAD components be implemented as separate tasks under the host OS.

Implementation Alternatives for Multiple-Host Systems

As previously noted, an appreciation of IPAD as a multiple-user distributed computing system requires an understanding of the scheme for implementing IPAD on each host computing system (ref. 9).

As a fully distributed computing system, IPAD is based on a direct data transfer bus-type network. Data is passed directly between network computers, and network control is distributed throughout the network. A "kernel" network control function, which handles the distribution of functions and data, is replicated on each network computer.

In any computing system, computing tasks request services and resources, which are provided through the host operating system. They include input, output, and file operations; process control; intertask communication; etc. In single-host computing systems, all of the services and resources are locally available. In distributed computing systems, they are supplied by computers throughout the network. The computing tasks in such systems request services and resources just as they do in nondistributed systems, and these requests are made using the same interface on any computer in the network. It is the responsibility of the OS on each computer to obtain requested services and resources regardless of location. The configuration of the distributed

computing system and the location of services and resources must be transparent to the computing tasks.

Figure 5 shows two interconnected host computers. Task A requests a service. The local OS, to which task A makes the request (line 1), interprets the request, determines that task B provides the service, locates task B, packages the request, and sends it (line 2) to the computer that contains task B. The OS on host 2 receives the message from host 1, acknowledges receipt, recognizes the request for a service provided by task B, saves the location of task A, and passes the request to task B (line 3). After completing its work, task B requests that results and status be returned to task A (line 4). The local OS interprets the request, recalls the location of task A, and packages and sends the results and status to the computer containing task A (line 5). The OS on host 1 receives and acknowledges the message containing results and status, interprets the message, and passes it to task A (line 6).

A fully distributed system also provides for the distribution of data. The location of files is kept in the system configuration tables. Tasks request the desired operations on the data base from the data management system, using logical file names. The data management system processes these and requests the OS to perform the desired operations on these logical files. The OS maps the logical file name, locates the file, and initiates the sequence of operations required to perform the file operation. If the file is local, it will initiate the appropriate file operation; otherwise, it formulates a message containing the request and follows the scenario previously described.

The preceding scenarios specify functions that must be provided by the OS in a distributed computing system, together with data needed by the system and certain system attributes.

Each OS in a distributed computing system must receive and interpret requests for services; locate services; package requests into messages; send, receive, acknowledge, and interpret messages; and direct messages to the appropriate tasks.

Locating a service (and files) requires access to timely information giving the location of all functions and data. This information must be updated whenever an event occurs that changes the configuration of the network or the status of functions and data. Each OS must have access to this information and provide information on the status of its functions and data when requested and when the status changes.

Operating systems in distributed computing systems must be able to communicate with each other. This requires a standard communication protocol for messages and data. Since computers in a distributed computing system may use different representations

for internal data, each must provide translators to and from the network standard data format. Each OS must receive and recognize requests for services provided throughout the distributed computing system. The interface to these services (calling sequence and return) must be identical on each computer.

None of the IPAD host computing systems are distributed computing systems. The IPAD implementation alternatives are concepts for providing the necessary services. Three major implementation concepts were considered.

1. Incorporate the functions necessary to provide distributed functions and data into each host computer OS:

 Case A: Implement all major IPAD components as a single task under the control of the OS.

 Case B: Implement the IPAD components as individual tasks under the control of the OS.

2. Implement IPAD as a single task under the control of the OS (including IPEX and the other major components) and develop IPEX to provide the functions and interfaces necessary for distributed computing.

3. Implement each major IPAD component (including IPEX) as individual tasks under the control of the OS and develop IPEX to provide the functions and interfaces necessary for distributed computing.

PRELIMINARY DESIGN AND FUTURE DIRECTIONS

The content of this paper has reflected full IPAD level II, at the time it was presented to NASA and ITAB during preliminary design review (PDR) in September 1978. Subsequent to this evaluation, it was decided by NASA and ITAB that specific areas of the technology covered during the PDR be further developed. This meant a reduction of effort in the following components: IPEX, user interface, user functions, and graphics. With the major thrust then to be directed toward the development of a prototype IPAD (first level) specifically in the area of data management and IPIP, the IPAD data base management system.

A discussion of the effort, which has extended since September 1978 until the contract termination date of February 1981, is included in proceedings and papers presented at the IPAD National Symposium held in Denver, Colorado, September 17-19, 1980 (refs. 10, 11, 12, 13, 14, and 15).

CONCLUDING REMARKS

The preliminary design of full IPAD has proven that the complexities of managing an engineering design project can be greatly enhanced by the use of a comprehensive user-oriented computer system.

The capabilities and functions provided to the design engineer within a CBEC are wide and varied, thus allowing both technical and project management within the engineering complex.

From a technical standpoint, the state of the art in computing is pressed to the limit in all areas of functionality to support the man/machine dialog necessary to promote and increase productivity throughout the engineering design process. See figure 6.

REFERENCES

1. IPAD Computing Staff, IPAD Document, D6-IPAD-70036-D-1, IPAD System Design Overview, August 1978.

2. IPAD Computing Staff, IPAD Document, D6-IPAD-70036-D-2, User Interface Preliminary Design, March 1979.

3. IPAD Computing Staff, IPAD Document, D6-IPAD-70036-D-9, Level II IPAD, January 1980.

4. IPAD Computing Staff, IPAD Document, D6-IPAD-70036-D-3, IPAD Evaluations and Alternatives, August 1, 1978.

5. IPAD Computing Staff, IPAD Document, D6-IPAD-70036-D-4, IPEX Preliminary Design, January 1979.

6. IPAD Computing Staff, IPAD Document, D6-IPAD-70036-D-6, IPAD Graphics, March 1979.

7. IPAD Computing Staff, IPAD Document, D6-IPAD-70036-D-7, IPAD Geometry, March 1979.

8. IPAD Computing Staff, IPAD Document, D6-IPAD-70036-D-8, User View of IPAD, May 1979.

9. Lixvar, John P: A Boeing IPAD System, DSG-Report-008, May 1979.

10. Southall, John W.: Requirements for Company-Wide Management of Engineering Information. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 7 of this compilation.)

11. Tanner, J. G., et al.: Executive and Communications Services to Support the IPAD Environment. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 10 of this compilation.)

12. Johnson, H. R., et al.: An Engineering Data Management System for IPAD. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 11 of this compilation.)

13. Dube, R. Peter, et al.: An Approach for Management of Geometry Data. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 12 of this compilation.)

14. Bryant, Walter A.; and Crowell, Harold A.: User Involvement in IPAD Software Development. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 13 of this compilation.)

15. Miller, Ralph E., Jr.: IPAD Products and Implications for the Future. IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 14 of this compilation.)

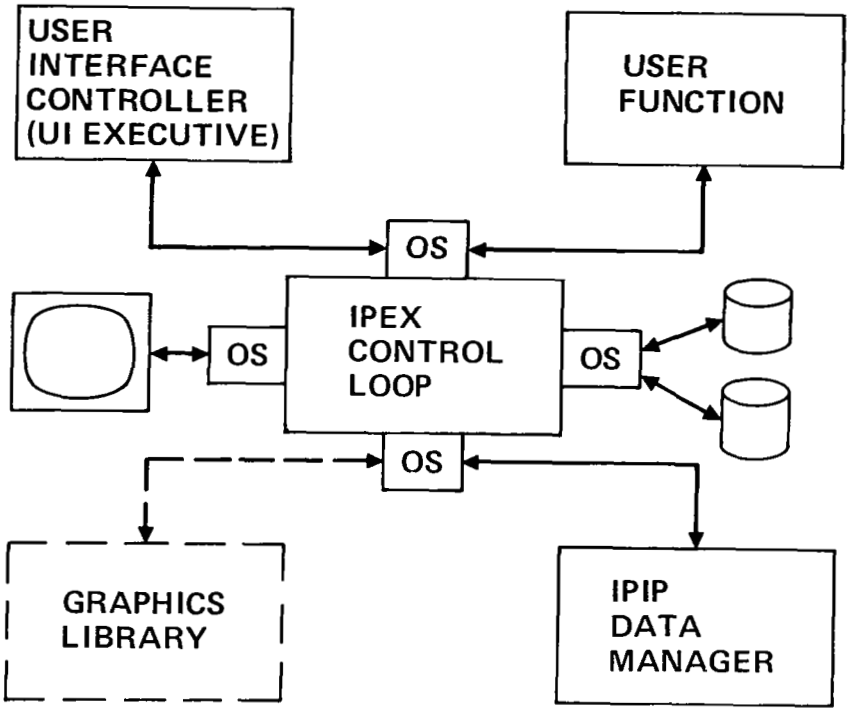


Figure 1.- IPAD level II representation.

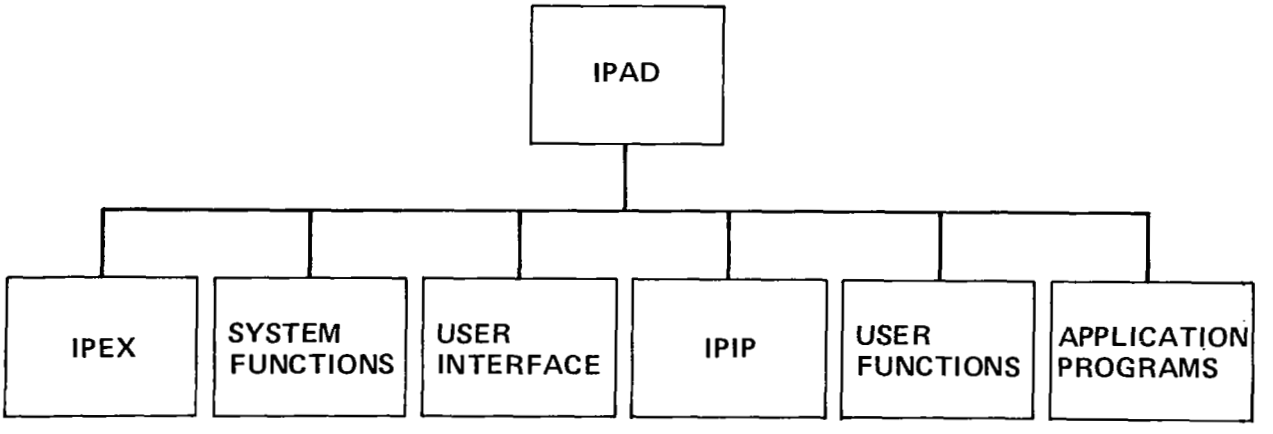


Figure 2.- Six major components sets.

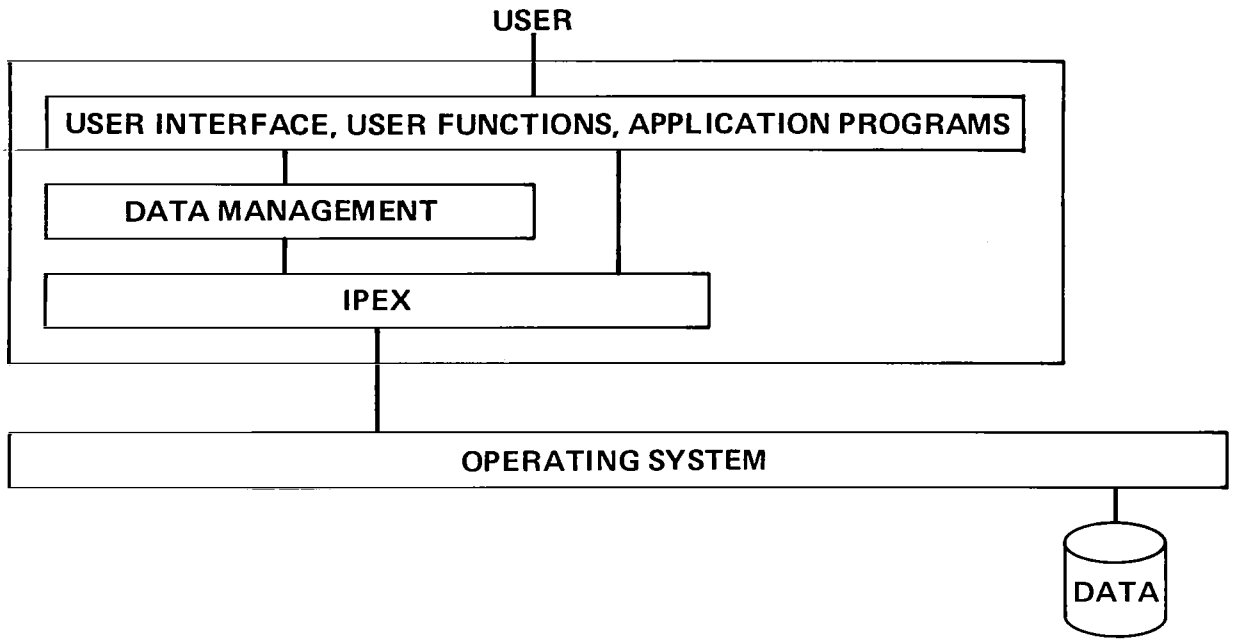


Figure 3.- Access relationships.

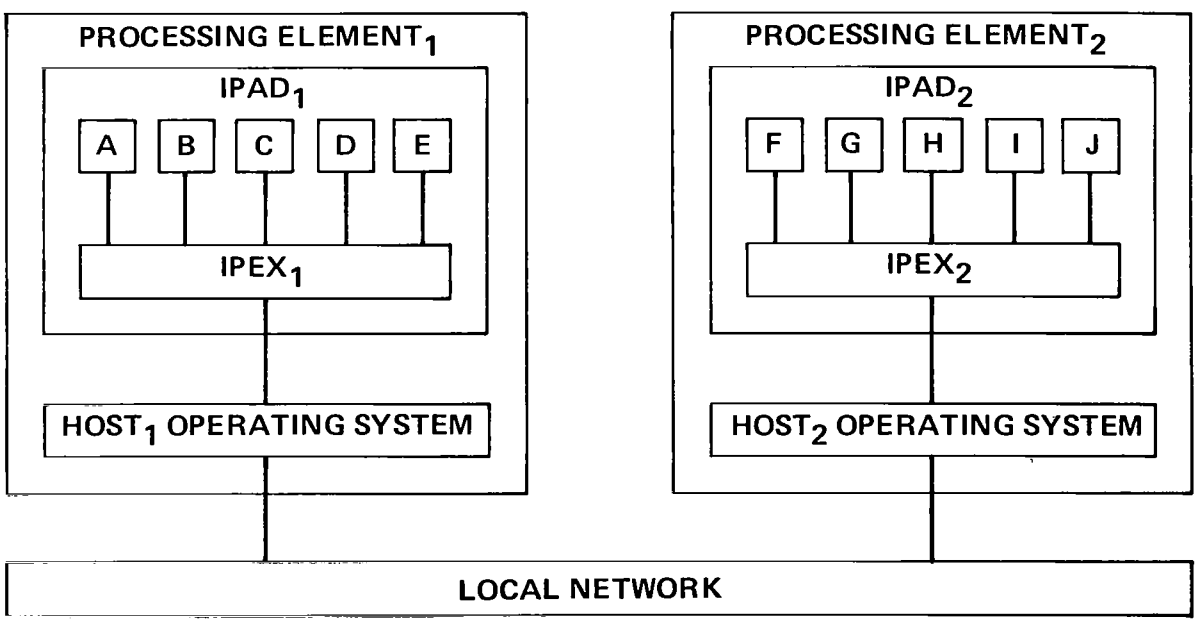


Figure 4.- Two-host configuration.

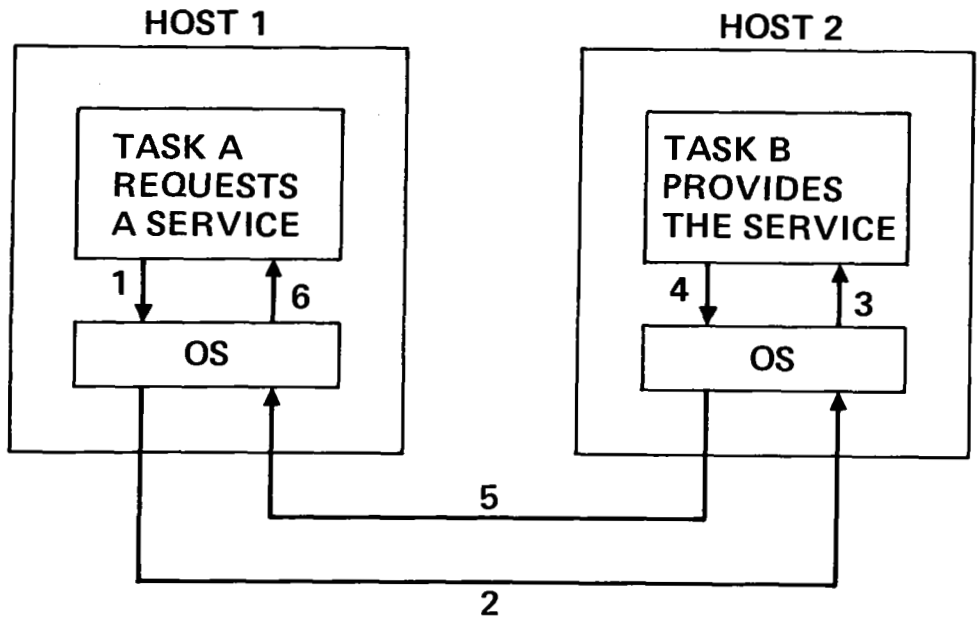


Figure 5.- Services in a distributed computing system.

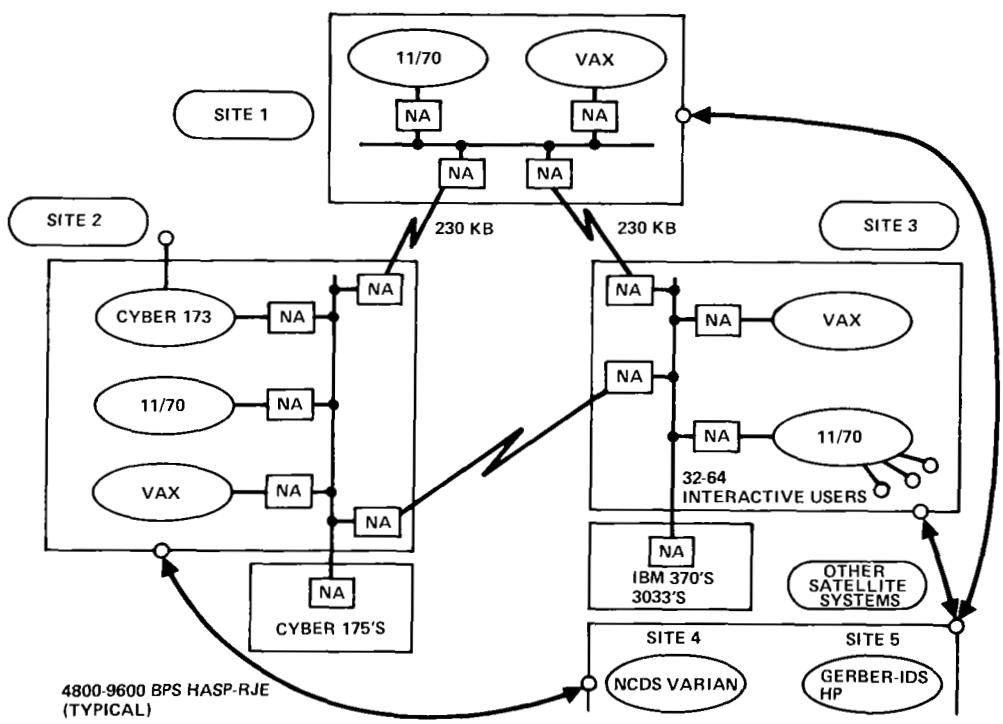


Figure 6.- IPAD network.

EXECUTIVE AND COMMUNICATIONS SERVICES TO SUPPORT THE IPAD ENVIRONMENT

J. G. Tanner, D. M. Kirkwood, F. M. Ives
Boeing Computer Services Company

SUMMARY

The principal purposes of the prototype executive software are to provide a system independent interface to the underlying host system and to allow for extension to full IPAD executive services as described in the preliminary design. A basic set of functions is included in the prototype to meet the requirements of the other components of the prototype, principally IPIP, the IPAD data management system. The functions were chosen so that they would be readily built on any of the proposed host systems with minimal redesign and execution overhead. The functions fall into five categories: access to host data, access to data files, access to communication services, data transformation, and instrumentation for performance measurement.

Communication services provide message delivery between processes in a network of heterogeneous computers. Data transformation services and communication services ensure data type validity and data integrity of messages exchanged between processes.

In the prototype, communication services use a high-speed local network to connect the computers; this does not preclude the use of other communication subnets such as a public packet-switched network. Extendability is important for both executive services and communications services.

INTRODUCTION

The following discussion is necessarily limited to a brief overview of the IPAD system and IPAD executive (IPEX) design; for a more extensive description, the reader is referred to the IPAD Level Two Design document (ref. 1). Initially, we are building a prototype system that is a subset of the preliminary design review (PDR) concept and concentrating effort in the areas of data base management, geometry, and communications. This section also discusses the strategy for making the transition from the prototype to full IPEX. The remaining sections discuss the prototype design in more detail and explain the use of a high-order language (HOL) in the design and implementation of the prototype system.

IPAD System Overview

The principal purpose of the IPAD system is to support integration of engineering and project management activities for large engineering projects. The system is made up of (1) a user interface, (2) a data base management system, (3) an executive, (4) system functions, (5) user functions, and (6) application programs.

Geometry is an integral part of IPAD. Geometry manipulation and display capabilities are produced by a combination of data base management functions, user functions, user interface, and graphics.

Engineering requirements for the system are aimed at designing and building a tool that will increase the productivity of the engineers and managers on such projects (refs. 2, 3, and 4). Most of the requirements affected the design and implementation of the executive software either directly or indirectly. Some requirements that directly affect executive software are the following:

1. The system should be built with few or no modifications to the host operating system(s).

2. The software should run in either a single or multiple host environment.

3. The multiple host system could and probably would be a multiple manufacturer system.

4. The system architecture and the nature of the host(s) in the system should be transparent to the user (e.g., a uniform accounting system) except for considerations such as the precision and accuracy of numerical results.

5. The nonexecutive IPAD software should be as portable as possible, with a minimum of host system dependencies.

6. Wherever and whenever possible, national and international standards should be adhered to.

7. The system should support communication with non-IPAD systems and permit access to design data under IPAD control from non-IPAD systems.

8. The system should permit transfer of program modules and data between IPAD installations.

9. The system should provide controlled access to resources and data by processes within and outside an IPAD installation based on a user's capabilities.

As a consequence of these requirements, the IPAD executive (IPEX) is designed as a subsystem to support IPAD activities and tasks in a distributed, heterogeneous environment. IPEX will also function in a nondistributed, or homogeneous, environment. The logical view of IPAD components is shown in figure 1, and the relationships of IPAD components to each other and to the host operating system are shown in figure 2.

IPEX Design Overview

IPEX is designed to provide a uniform view of the underlying computer system(s) to the other components of IPAD. In addition, it provides controlled access to resources independent of the location of the requestor and resource (ref 5).

Requests are made of IPEX by building a service request block and asking the host operating system to pass this block to IPEX. IPEX assigns a request process to this service request block, and when the request has been processed, IPEX asks the host operating system to pass a service completion block to IPEX. IPEX assigns a request process to this service request block and, when the request has been processed, IPEX asks the host operating system to pass a service completion block to the task that requested service. This process is depicted in figure 3.

Internally, IPEX is organized as a group of independently scheduled cooperating processes that fall into one of three groups: request processes, server processes, or function processes.

There is a request process for each request made of IPEX. The request process decodes the request and generates the necessary internal requests to other IPEX components, such as server and function processes. The relationships between internal processes, a requesting task, and the host operating system are shown in figure 4.

In most cases the internal requests generated by the request process are sent to server processes. These processes interface with the host system to accomplish such host-dependent activity as file access, I/O device access, and task control. The reason for dividing request servicing in this manner is that the host system involved in servicing the request may not be the host on which the request was made. Request processes interface with the requesting task, and server processes interface with the target host.

IPEX, as described above, was removed from the system when we decided to build a reduced-function prototype. We then had to design operating system interface software that would satisfy a subset of the full IPEX functions and evolve into the full IPEX design.

Strategy of Evolution

One of the primary concerns in defining the prototype was that it had to be built within time and money constraints. This concern led to limiting the design to the solution of a subset of the problems addressed by the full IPAD design. To concentrate our efforts still further, we focused our attention on the areas of (1) data base management, (2) geometry, and (3) data communication.

Regardless of other design choices for the prototype, it is important to insulate the user and nonexecutive IPAD software from changes in the lower-level support software. This approach also allows some or all of the identified IPEX functions to migrate into the host system software, which is vendor-supplied and -maintained. The migration process will not affect the user's view of the system. Some of the major considerations in the design of the executive software were the following:

1. A uniform user interface must be established.

2. The evolution from the prototype software to full IPEX should be transparent to the user.

3. IPEX would not exist as a separate task and address space.

4. The system should provide some form of task-to-task communication.

5. Functions and resources would not be implicitly or transparently distributed.

6. The system configuration would consist of two nodes: CDC CYBER 172 (NOS) and a DEC PDP 11/70 (IAS). These systems were later changed to a CDC CYBER 720 (NOS) and a DEC VAX11/780 (VMS).

7. The design should not preclude software migration to an IBM host.

In the design for full IPAD, system services (e.g., file access) are to be provided by a central IPAD executive (IPEX), to be shared by all users; but, for the prototype IPAD, system services will be provided by multiple copies of IPEX service routines (IPSRs), one set to each task. These IPSRs will combine the functions of the request processes, server processes, and function processes of IPEX.

IPEX SERVICE ROUTINES

The IPSRs are the IPAD application's and IPIP programmer's views of the host system environment. The functions available constitute a subset of the functions provided by the host system. They are a subset because they form an intersection of capabilities of the possible host systems. No function or capability is provided in IPSRs that is not mapped easily to existing host system services.

First Stage of Evolution

The IPSRs will reside in the same address space as the requesting task and fall into these functional categories:

File I/O
Network I/O
Access to task and host data
Task control
Instrumentation primitives for gathering performance data

Device I/O was not considered for the prototype, and terminal I/O is supported adequately by the programming language. In addition, data transformation software is used in conjunction with network I/O to maintain data compatibility between processes. Because the IPSRs reside in the same address space as the users software, access to host system services occurs by calls and not by passing request blocks to IPEX.

This first stage of evolution provides access to local host system resources. Access to remote system resources must be done explicitly by using network I/O facilities and therefore assumes a knowledge of resource location.

The Migration Path

The move from the prototype IPSRs to full IPEX will be a stepwise process, the first step being the building of the IPSRs for the prototype. The next step is to build a basic version of the central shared IPEX as designed at PDR. Internal elements of this basic version will include the control loop and the request processes. Migration will proceed by adding server and function processes to IPEX to support the functions currently supported by IPSRs. During this migration stage, IPSR-supported functions and IPEX-supported functions will coexist.

Once support for all the prototype functions has been assumed by IPEX, experimentation with new functions and capabilities can be done by adding server and function processes to IPEX. One of

the primary areas of interest is resource sharing and access to remote resources. In general, we are interested in the distribution of both data and functions in a heterogeneous computer network.

The Basic Divisions of Service

There are five basic divisions of service supported by IPSRs, and these are the fundamental functions needed by IPAD applications and IPIP. We emphasized the three areas of file I/O, network I/O, and instrumentation primitives, because these were considered most critical to the development of other IPAD components. The two others--(1) task control and (2) access to task and host data--are provided but not supported to the same extent as the three primary areas. All the areas are discussed briefly in this section, and detailed discussions of file I/O, network I/O, data transformation, and instrumentation and performance measurement occur in later sections of this paper.

IPAD Host Access System (IHAS).- The IPAD host access system (IHAS) must provide identical host computer access/control services to any IPAD task executing on any of the host computers in the IPAD computing complex. To do this, IHAS provides a set of routines (IPSRs) that enable an IPAD task to access and control its underlying host computer system. IHAS functions allow an IPAD task to:

1. Obtain host system characteristic data
2. Determine its own execution status
3. Suspend its own execution
4. Set its own conditions for continuing execution
5. Obtain time and date

An IPAD task would invoke the HSTDATA function to determine which of the host computers in the IPAD computing complex it is executing on. It will use these host-characteristic data to set up file record lengths and extents, character set usage, word size, etc.

The IPAD task can suspend its own execution with the TSKSPND function after determining its execution status with the TSKSTAT function. If an error occurs during its execution, the IPAD task can handle this itself with the TSKERR function.

Time and date maintained by the underlying host computer's operating system are accessible through the HSTTIME and HSTDATE functions.

IPAD File Control System (IFCS).- IFCS maps its host-independent functions onto those of the underlying host operating system. Using these functions, an IPAD task may create new files; extend, shorten, or delete existing files; and read, write, and update records within a file. IFCS is discussed in more detail in a following section, and figure 5 summarizes functions and their parameters.

Network Services.- Network services provide the IPAD task access to all available communications services. For the prototype, this consists of a high-speed local network of two computers. Network functions allow an IPAD task to connect to and disconnect from the network and to send and receive messages to and from other tasks connected to the network.

Network services are discussed in more detail in a later section of this paper, and figure 6 summarizes functions and their parameters.

Data Transformation.- Data transformation services facilitate process-to-process communication in a heterogeneous environment. Functions provide both item and record transformations between machine representation and standard representation. Data transformation services and standard representation are discussed in more detail in a later section of this paper.

Instrumentation and Performance Measurement.- Instrumentation is extremely important in tracking and altering the performance of a computing system, especially a new system. In addition, it helps verify the model of the system. The verified model can then be used with greater confidence to predict the performance results of a change to the system. Functions exist to record counts and times and to trace data at several levels of detail.

IPAD FILE CONTROL SYSTEM (IFCS)

The IPAD control system (IFCS) must provide identical file-type services to any IPAD task executing on any of the host computers in the IPAD computing complex. To do this IFCS provides a set of routines (IPSRs) that enable an IPAD task to access IPAD files using the facilities of the host computer system it is executing on. IFCS enables an IPAD task to:

1. Create new files

2. Extend, shorten, or delete existing files

3. Read, write, and update records within a file

Through file access functions and their associated parameters, an IPAD task transmits file access commands to the underlying host computer system and receives completion status information and data in return.

With IFCS, files are constructed as a one-dimensional array (or sequence) of records. This is achieved through exclusive use of the host computer's "direct" form of file organization. A sequential or tape-like form of file organization is not used. The access methods supported by IFCS are sequential and random.

IFCS File Attributes

Each IFCS file has certain logical and physical characteristics. In the IPAD prototype environment, some of these characteristics, or attributes, are fixed. Files created by IFCS have the following restricted set of attributes:

1. All IPAD files reside on direct-access storage devices (disks). They are organized as a physical structure, wherein IFCS considers the file to be a one-dimensional array of fixed-size records. An access index is associated with each record in the file and has values ranging from one to the total number of records in the file.

2. A record is the fundamental unit of information in an IPAD file. IFCS provides access to records in files but does not support an awareness of logical relationships among records/information in IPAD files or any knowledge of record content. Each record is processed as a single, indivisible unit of data whose size is an integer multiple of the size of the host system's storage device block. Each time a record is accessed, this many blocks are transferred. IPAD tasks build individual records and pass them to IFCS for storage in an IPAD file or issue requests for IFCS to retrieve records from an IPAD file.

3. Each file has a unique name, which is used by the underlying host computer's file access method to identify the file in its directory. The file's name is coded in the host's character set and is therefore host-dependent.

4. All records in an IPAD file have the same fixed length, which may be different on different hosts due to the different

characteristics of the host computer's direct-access storage device.

5. IPAD files have a maximum size, depending on the host computer system.

6. Read-only access can be defined for an IPAD file.

IFCS File Access

An IFCS file can be accessed either randomly or sequentially. Random access is achieved by specifying the record's access index; then an individual record can be either read, using the READ function, or completely replaced, using the WRITE function.

The sequential access mode is used to build an IPAD file using the WRITE function and to access it using the READ function without needing to specify the record's index. Records are accessed in the order in which they were placed on the file; however, by using the POSITION function, the record to be next accessed can be other than the one next in sequence. This function can specify relative positioning (forward space and backward space) and absolute positioning (rewind to load point and space to a record specified by its index). An IPAD file can be made smaller either by positioning the file to the last meaningful record or by specifying the index of the record and then executing the TRUNCATE function. An IPAD file can be extended by using the POSITION and WRITE functions.

To access an existing IPAD file, an IPAD task must first connect to it using the OPEN function. An IPAD file can be created by using the OPEN function to define the file's attributes and to signal the creation of a new file. This causes the underlying host operating system to create an entry for the new file in its directory, allocate space, and position the file to the beginning of the allocated space. Repeated use of the WRITE function will then add records to the new file.

When file building or accessing is complete, the IPAD task disconnects from the file by using the CLOSE function.

An IPAD file can be removed from the host computer system's directory by using the DELETE function. At this time, allocated space may be returned to the host system for reuse.

Figure 5 summarizes the IFCS functions and their parameters.

NETWORK SERVICES

Network services provide communication between processes in the IPAD system. This is done in a unified way with a combination of hardware and software subsystems. The communications system is designed to interface with a variety of communication subnets while keeping the communications interface to the user constant and uniform. This approach is in keeping with the philosophy of insulating users from changes in the mechanics of communications and permits taking advantage of new communication networks and changing communication technology.

Communications Architecture

To further insulate ourselves from changes, we adopted the layered architecture proposed by the American National Standard for Information Interchange (ANSII) and the International Standards Organization (ISO). The proposed architectural model contains the following seven levels (or layers) of control:

Level 7. Process control

Level 6. Presentation control

Level 5. Session control

Level 4. Transport control

Level 3. Network control

Level 2. Link control

Level 1. Physical control

The communications subnet supports physical and link control functions as well as some network control functions. The communications control program (CCP) software supports session, transport, and the remaining network control functions. The relationships between these layers and among their peers are depicted in Figure 7.

Network control is transport control's interface to the communications subnet hardware. The network control interface supplies functions, status, and operational behavior characteristic of this type of network. This isolates transport control from knowledge of hardware details, thus freeing it to concern itself with functions common to all networks. Some of these functions are routing, flow control, end-to-end acknowledgements, segmentation, blocking, and message

retransmission. Transport control handles external communications or internode data transfers.

Session control handles internal communications or intranode data transfers, performs initial routing to determine if it requires the services of transport control or if it should handle the transfer itself, and interfaces with the user process to move messages to and from other processes (e.g., other user processes or server processes).

Server processes provide special functions or services and are usually associated with a particular resource or resource class. Data transfer between sessions is transparent, which implies that session control and lower levels of the architecture transmit the data without interpretation. Session control and lower levels may add protocol information in the form of header data, but they do not analyze or alter data from a higher level.

Presentation services are primarily concerned with transformation of data between a machine-and-language-dependent representation and a machine-and-language-independent (or standard) representation. These services are described in detail in a later section of this paper.

We adopted a layered architecture to insulate the user from changes in lower-level functions and communications technology. To minimize the effects of expansion and contraction of the network and changes in its topology, we adopted a three-level heirarchical addressing scheme to name processes in the network. The network is viewed as a four-level heirarchy consisting of (1) processes, which reside on (2) nodes, which are grouped into (3) clusters, which are interconnected to form the (4) network.

This is a conceptual framework and is not intended to specify the physical interconnection of the hardware making up the network.

The name by which a process is known in the network is called its network name and consists (1) a cluster name, identifying the cluster containing the process, (2) a node name, identifying the node containing the process, and (3) a process name, identifying the process within the node.

This addressing structure is analogous to a telephone number containing an area code, an exchange code, and a four-digit subscriber code. An example of process addressing in a three-cluster network is shown in figure 8.

It is obvious that within the node, a process name must be unique; within the cluster, the node name must be unique; and within the network, the cluster name must be unique. In order that special processes can be contacted without knowing their

specific nodes or clusters, some process names are unique within their entire cluster or within the entire network. Each process name has a specified degree of uniqueness indicated by its class. The process name classes are (1) node class (the process name is unique within the node), (2) cluster class (the process name is unique within the cluster), and (3) network class (the process name is unique within the network).

There is more discussion of process name classes and addressing processes in a later section of this paper.

Hardware Environment

The communication subnet is a bus architecture network consisting of two microprocessor-based network adapters connected by a standard coaxial cable trunk. The network adapters interface a host computer channel with up to four trunks. Adapters transfer data over any one trunk at a rate of 50 megabits per second. This data rate can be maintained in a 16-adapter configuration with maximum trunk lengths of 1000 feet. Data rates decrease as the number of adapters or cable lengths increase.

This is a local high-speed network operating at channel rates. As a consequence, this network supports communication with peripheral devices as well as other CPUs. Our network is not configured to do this, and currently our software does not support this feature. Our current hardware configuration is shown in figure 9 and includes a CDC CYBER 720 running NOS and a DEC VAX 11/780 running VMS. These two systems are connected by two Network Systems Corporation (NSC) network adapters (Hyperchannels) and a single coaxial cable trunk (Hypertrunk).

Software Environment

The logical organization or communications architecture has been described already. Physically, the software is organized into two components. Network I/O services are part of the user's address space and execute when called by the user at various functional entry points. The communications control program (CCP) occupies a separate address space and executes as a separate process, asynchronous to the user's process.

The network I/O services maintain the user interface to session control, the highest layer of the architecture that concerns the software. The network I/O services, in combination with the CCP, provide communication between processes within the network. As such, the primary services are those that allow one process to send a message and another to receive it. The sending process is sometimes referred to as the source process and the receiving process as the destination process. A process sending a

message must be able to specify the destination. A process willing to receive a message must be able to specify which process or processes it will accept as sources. For this reason all processes wanting to communicate over the network must have names that are known to the network. The form of network names has been discussed under architecture.

A process network name is associated with the process when it connects to the network. Until a connect operation is performed, the process is not known to the network software and cannot use any of the network facilities. A process can terminate its connection with the network software by a dissconnect function.

Once connected to the network software, the process may send and receive messages. These two operations are the main purpose of the software. The remaining services (check, wait, and cancel) are merely for the purpose of augmenting the operation of sends and receives. Send and receive operations may be performed synchronously (in which case the process is suspended until the operation has completed) or asynchronously (in which case the process is allowed to continue with other work while the network software does the operation).

In the procedure descriptions in the next section, reference is made to incomplete requests. A request is an operation requested by a process and is incomplete as long as the network software maintains information on the request. Synchronous requests are generally completed before the requesting process is reactivated. The only way for a process to complete an incomplete request is to determine when the network software is ready to complete the request and, at that time, reissue the request, perhaps with different parameters. The check and wait operations can be used to ascertain whether a request is ready to complete. One of the parameters in the send and receive requests determines whether the request is new or an attempt to complete an old one. The cancel command can be used at any time to cancel an incomplete request (or for that matter, to cancel all incomplete requests at once).

The one case in which a synchronous operation can be incomplete when control is returned to the process is when a process asks to receive a message of length n and a message of length m ($m > n$) is sent. In this case the first n units of the message are delivered, the process is reactivated, and the request is incomplete. The process can then decide whether to reissue the request for the remainder of the message or cancel the request.

The remaining terms to be discussed in this section are port number and time out. When a process sends a message, it specifies both the network name of the destination process and a port number. Conceptually, each port number represents a potential line of communication with the destination process, and several

communications can be carried on in parallel if they are addressed to separate parts. The port number allows the receiving process to be selective about which messages it receives. It may specify that it is only willing to accept messages from a certain process or group of processes by the way it specifes the source process network name, or it can specify that it is only willing to accept messages sent to a specific port, or it can combine these selective capabilities.

Time out refers to the situation in which a request has not been satisfied within some time limit specified by the requesting process. When this happens the request "times out," that is, the network software terminates processing of the request and either completes the request (synchronous requests) or marks the request as ready for completion (asynchronous request). When the request completes, an appropriate error code is returned to the user.

When called, the network I/O services, in the user's address, package requests and pass them to the CCP for service. Network control, transport control, and the remaining session control functions are implemented in the CCP. The organization of the CCP and its relationship to the user processes, the node, and the communications subnet are shown in figure 10.

Network I/O Interface Procedures

Connect to Network

This procedure establishes (1) the process as a network user and (2) its network name.

escnet (name, class, return code)

name — This refers to the name by which the process is known within the network. Only the process name is specified by the user; the network software will fill in the cluster and node fields. The user can request that the network software also fill in the process field with a node-unique name by leaving that field blank.

class — Refers to one of network, cluster, or node class. A process must have special privileges in order to specify cluster or network class.

Disconnect From Network

This procedure removes the process as a network user. The network will terminate any incomplete requests involving this process, as either source or destination, with appropriate error messages.

esdnet (return code)

Send a Message

This procedure sends a message to a destination process. If the request is synchronous, the process suspends until the network software is satisfied that the destination process has received the message or until an error condition is detected or until the request times out. When the process is reactivated the request will have been completed. If the request is asynchronous, it will complete only if the message can be delivered immediately (i.e., if there is a receive pending for the message and the entire message can be delivered without further action on the part of the receiving process). If the asynchronous send request is not immediately satisfiable, the network software establishes it as an incomplete request and returns an identifying number to the calling process.

The above specifies the function of the send procedure if the id parameter is 0, indicating a new request. If the id is not zero, then this is a reissued request for the incomplete send identified by that id, in which case the network software checks the current status of the incomplete request and, depending on whether the incomplete request is ready to complete and the reissued send is synchronous or asynchronous, proceeds as specified above.

essend (name, port, units, length, message, mode, id, time, return code)

name Name specifies the destination process. The process field must be the process name of the destination process. The cluster and node fields may contain names or the word "local" or blanks. The implications of the various combinations are:

Cluster	Node	Implications
blank	blank	the process name is network unique
blank	"local"	not allowed
blank	name	not allowed
"local"	blank	same cluster as sending process process name is cluster unique
"local"	"local"	same node as sending process
"local"	name	same cluster as sending process
name	blank	name is cluster-unique
name	local	not allowed
name	name	completely specified name

The send request causes the actual cluster and node names of the receiving process to be filled in if the fields contain blanks or the word "local."



port — Port is the port number the message is being sent to. Unless the sender and receiver have agreed to some other port number by convention or negotiation, the standard port number is 1.

units — The number of bits per message unit

length — The length of the message in units. The number of bits in a message is computed by length x units.

message — The message to be sent

mode — synchronous or asynchronous

id — Zero (0) implies this is a new request. If the send request has not completed when send returns, this parameter contains the id of the incomplete request. Nonzero implies that this is a reissued request (an attempt to complete the incomplete send request identified by id).

time — The time (in seconds) before this request should time out. If the request is not satisfied within the specified time limit, the request will time out. This parameter is ignored on reissued requests.

Receive a Message

This procedure receives a message from a source process. There are three modes for receive requests:

synchronous — Wait for the request to be satisfied (or time out or error) before returning to user process.

asynchronous — If the request is immediately satisfiable, perform it, otherwise establish it as an incomplete request and return.

conditional — If the request is immediately satisfiable, perform it. Otherwise return without doing anything.

The reissued forms of these requests (id parameter nonzero) behave pretty much as the initial request, except that they reference an incomplete request. Since reissued requests reference already established incomplete requests, a reissued conditional receive is identical to a reissued asynchronous receive.

Receives are unique in that they may be partially satisfied if the message sent is longer than the length specified in the

receive. In this event (for all three modes, in both the initial and reissued request cases), the procedure delivers the portion of the message specified by length, establishes the request as an incomplete request (if it is not already), and returns to the calling process. The process now has the option of using reissued request(s) to pick up the remainder of the message or cancelling the request.

esrecu (name, port, units, length, message, node, id, time, return code)

name Name specifies the acceptable source processes. The fields may contain valid names, blanks, or the word "local." The implications of the various combinations are:

cluster	node	process	meaning
blank	blank	blank	any process
blank	blank	name	any process with specified process name
name	blank	blank	any process in specified cluster
name	blank	name	any process in specifed cluster with specifed process name
name	name	blank	any process on specited node
name	name	name	completely specified name

"Local," used in place of a cluster name means "the cluster containing the calling process."

"Local," used in place of both the cluster and node names, means "the node containing the calling process."

Other combinations are invalid. The receive procedure replaces blank and "local" fields by the actual cluster, node, and/or process names of the source process.

port The port number at which a message will be received; it is a further constraint on acceptable messages. Specifying a port number of 0 (zero) indicates the process will receive at any port. In this case, the actual port number to which the message was sent is returned by the receive procedure.

units The number of bits per message unit

length The length of the message in units. The number of bits in a message is computed by length x units.

message — the received message

mode — synchronous, asynchronous, or conditional

id — Zero (0) implies that this is a new request. If this request is established as incomplete when the receive procedure returns, the receive procedure places the id of the incomplete request in this parameter. Nonzero implies that this is a reissued request (an attempt to complete the incomplete receive request identified by id).

time — The time in seconds before this request should time out. If the request is not satisfied within the specified time limit, the request will time out. (This parameter is ignored on reissued requests.)

Check an Incomplete Request

This procedure checks to see if a specified incomplete request is ready to complete.

eschek (id, return code)

id — the id of an incomplete send or receive request

return code — indicates whether the specified incomplete request is ready to complete

Wait for a Request to Become Ready to Complete

This procedure waits for a specified request to be ready to complete and waits only for a specified time.

eswait (id, time, return code)

id — Zero (0) means wait for any incomplete request to become ready to complete. If one does within the specified time limit, its id is returned in this parameter.

Nonzero means wait for the incomplete request identified by this id to become ready to complete.

time — If the specified incomplete request (or any if none was specified) does not become ready to complete within this time limit, stop waiting.

return code — indicates whether wait was successful or timed out

Cancel a Request

This procedure cancels an incomplete request.

escncl (id, return code)

id Refers to id of the incomplete request that is to be cancelled. If id is zero, all incomplete requests issued by this process are cancelled.

DATA TRANSFORMATION

Data transformation is the translation of data values from one representation to another. In a heterogeneous environment such as the IPAD local network, data transformation is necessary for accurate communication of data values. The following sections discuss the need for data transformation, the nature of the data transformation problem, the general approach to a solution adopted on the IPAD project, and the specific IPAD software developed to handle data transformations.

Information Transfer in a Heterogeneous Network

In an integrated system, various parts of a problem are worked on by different processes that must coordinate their activity. (A process is the execution of a program.) This coordination usually involves information (such as the IPAD data base) that is available to all of the processes and manipulated by many of them. In addition, some information is transferred between concurrently executing processes. This active transfer of information is represented by the interprocess communication (network) software in IPAD.

Transfer of information between processes in the IPAD system, whether through the data base or interprocess communication, is complicated by the heterogeneous nature of IPAD. The IPAD local network interconnects a variety of computers. While the IPAD system software is implemented in one language, the user programs being integrated into IPAD are not constrained to this language. Thus, a process that produces a given data value may be implemented in a different language and may run on a different machine than one that needs to access that value.

The representation of a given data value or group of data values will vary from machine to machine. Even on a given machine different compilers choose different ways to represent a given data value. If IPAD software is to guarantee that a data value produced by one process is correctly presented to processes needing to access that value, IPAD software must perform

translation between the representation used by the creating process and the various representations expected by the referencing processes.

The unacceptable alternative is that all programs agree on some machine-independent representation of data values to be transferred between processes. Since this representation would be incompatible with the machines' instruction sets (or at least most of them), manipulation of these values would have to be software-implemented and inefficient. An example would be comparing two integers received from another process in the standard form. This comparison would have to be handled by a subroutine call instead of by using a simple comparison expression, because the expression evaluation would use machine instructions, which expect the integers to be in machine format.

Variation in Data Representation

Most machines and compilers distinguish between the basic data types--integer, floating point, character, and logical--and allow the programmer to access and manipulate these types individually or in arrays and records. Data transformation software is concerned with translating data values of these types between the representations of each of the machine and compiler combinations. The following paragraphs discuss the ways in which these representations vary.

Integer values are usually stored as binary numbers. The number of bits used is a function of the word size of the machine. Some machine architectures allow for more than one size of binary integer. The main variation other than the number of bits is the method of representing negative integers. The most prevalent methods are ones complement and twos complement.

Floating point ("real") values are generally of the form $m \times b^e$, where m is a signed integer or fraction, e is a signed integer, and b is the base. The representation involves an encoding for e and m. The base is understood, although the value of the understood base may vary between machine architectures. Bases of 2 and 16 are common. The representations of the exponent e vary both in the number of bits and in the method of representing the signed value of e. Generally the method of representing e is independent of the method chosen for representing integer values on the same machine and has been selected in conjunction with the design of the floating point hardware. On some machines m is an integer value, on others it is a fraction. The method for representing negative values for m also varies between machines and need not be consistent with representation of negative integers. Finally, the number of bits used to represent m varies between machines, with the total number of bits for m and e being a function of the machines word size.

Most machine architectures allow for more than one size of floating point value.

The primary encoding for characters is ASCII. Machines that have adopted this representation usually use eight bits to encode a character. CDC representation is called display code and uses six bits per character. IBM uses EBCDIC, which is an eight-bit encoding that differs markedly from ASCII. There is no simple transformation that maps any one of these encodings into the other.

Logical (Boolean) values are perhaps the simplest in that the only possibilities are true and false. One bit is adequate to represent a logical value. The representation of logical values is usually compiler-dependent rather than machine-architecture-dependent. Variations include positioning of the logical value within the machine word, the number of bits used to represent the value, and how true and false are represented (usually a question only if more than one bit is used).

Record and array structures vary due to word sizes and alignment requirements on different machines. Compilers usually allow (or make) different choices as to how Boolean and character values are to be packed within addressable memory units.

In general, there are few similarities between machine repesentations. Transformation to (from) a host machine's representation from (to) a foreign representation invariably requires treating the foreign representation as a bit string and disassembling (assembling) that representation at the bit level.

Network Standard Representation

One of the earliest decisions concerning data transformation in IPAD was that every transformation between machine repesentations would be performed as a two-step process. The data is first translated from the source machine representation into an intermediate form called network standard representation (NSR). This transformation is performed on the source machine. The data is then translated on the destination machine from NSR to the destination machine representation. The alternative, translating data directly between source machine representation and destination machine representation, would require two sets of transformation routines for each of the n(n-1) pairs of representations in a network with n machine types. Adding an n+1st machine type would require writing 2n new sets of transformation routines. If an intermediate representation is used, only 2n sets of transformation routines are required for all transformations, and adding an n+1st machine type only requires writing two new sets of transformation routines, one set

for translating from the new machine representation to NSR and one set for translating from NSR to the new machine representation.

Having settled on a standard representation, the next decision was on the form of this standard. Arguments can be made against using one of the machine representations as a standard, and this is a tempting choice if one of the machine types plays a central role in the current network configuration. The strongest argument is the loss of information that could result in translating through such an NSR. A value that is representable in both the source and destination machine representations may not be representable in the chosen standard. There is no one machine representation that is a superset of all the others in terms of representable values. Sufficiency, then, is a reqirement which must be met by the NSR. Any value representable on one of the machines in the network must be representable in the standard. Two other desirable characteristics of an NSR are compactness and ease of conversion to and from the chosen representation.

The NSR designed for IPAD has variable-length integer and real representations, which provides both for compactness (since it need only have enough bits for the desired precision) and for sufficiency (since it can have as many bits as necessary to achieve the precision of future machines added to the network). Character values are represented in eight-bit ASCII. This was chosen for the sake of sufficiency. There was no choice that would provide for ease of transformation between the various character representations. Logical values are represented by a single bit for the sake of compactness. The NSR representation of each of the basic data types is given in the following text.

BASIC DATA TYPES

The basic data types are real, integer, character, and logical. In SR (standard representation), each of these is parameterized. Data types, representations, and parameters are presented below.

Integer

characteristics:

used to represent integer values in the range $-2^p < i < 2^p$

parameters:

p

representation:

SR for integers uses ones complement binary notation. The all-ones bit pattern (-0 on some machines) is not used in SR.

size:

P + 1 bits

Real

characteristics:

Used to approximate real values. The actual values represented are a p bit fraction in the range $.5 \leq f < 1$, a sign, and an integer exponent in the range $-2^q < e < 2^q$. A value of zero is the only value for which $f < .5$.

parameters:

q
p

representation:

The exponent is represented by using ones complement binary notation. The sign (of the fraction) is represented by one bit (0=+, 1=-), and the fraction f is represented by p bits. Except for a value of f=0, the first fraction bit is always 1 (f is a normalized fraction). The value of zero is represented by all bits of the SR equal to zero.

size:

e	q+1 bits
sign	1 bit
f	p bits

Logical

characteristics:

used to represent true/false values

parameters:

n the number of true false values represented

representation:

one bit per value

0 = false

1 = true

size:

n bits

Character

characteristics:

used to represent character information

parameters:

n the number of characters

representation:

8-bit ASCII is used to encode characters

size:

8n bits

Bit

characteristics:

used to pass information without translation

parameters:

n number of bits

representation:

bits are copied directly from MR (machine representation) into SR and from SR into MR

size:

n bits

Records, Descriptors, and Field Modes

While there are data transformation routines that translate individual data items, most transformations are performed on records containing several data items of possibly different types. A machine record is a set of data in some machine-dependent

representation. A standard record is a set of data in the machine-independent NSR.

In order to perform a translation, it is necessary to know (1) the location of the source and target records, (2) what data types the records contain, and (3) their actual or intended representation (if that data type allows choices). This description of a record's contents is contained in any array of integers and is called a record descriptor.

Sending a message from a process on machine A to a process on machine B would usually require the following steps:

1. The originating process would pass the message and a record descriptor to the data transformation software, which would return the message in a standard record.

2. The originating process would then send this standard record to the destination process, using the network I/O routines.

3. The destination process would use the network I/O routines to receive the standard record and would then pass the standard record and a record descriptor to the data transformation software, which would return a machine record containing the message in machine representation for machine B.

It should be noted that, in this example, both processes needed to know in advance the appropriate record descriptor, i.e., the destination process had to know the format of the message. It is reasonable that, in some applications, the two processes would have a protocol for passing the record descriptor as part of the standard record. If such a protocol were used, the destination process would translate the descriptor first and then translate the rest of the record using this descriptor. The data transformation software allows such translation of partial records.

A record descriptor contains, for each field in a record, a field descriptor specifying the field type, any parameters associated with the machine, and standard representations of that field. To a large extent, a record descriptor is machine-independent, except that it is itself in a machine representation and that the parameters associated with the machine representations of a given field may vary from machine to machine. Usually, machine representation parameters can be chosen so that one record descriptor defines both the standard and machine representation on both the source and destination machines. To prevent the descriptor from being excessively large, there is a repeat field descriptor that indicates when a given field or set of fields is repeated. A record containing 100 (or 1000) real values requires only four field descriptors.

In early use of the data transformation software, it was discovered that it is occasionally desirable for certain fields to be extracted from the machine (standard) record and translated to form a smaller standard (machine) record. The reverse process of breaking a record up and inserting it piecewise into a larger record during translation is equally desirable.

The data transformation routines view the record descriptor as a description of a general record, certain fields of which may be missing from either the source or target record. Field masks are required for both the machine and standard records to specify which fields are present in those records. If no extraction or insertion is to be performed, both field masks would indicate that all fields are present.

Records, record descriptors, and field masks are the main entities involved in data transformation. A record descriptor and two field masks provide the data transformation routines with all of the information necessary to completely determine the representation of a given set of values in either machine or standard representation. This information and a machine (standard) record permit the creation of the selected fields of the corresponding standard (machine) record.

Data Transformation Software

This section includes definitions of data transformation terms and descriptions of the individual data transformation routines.

Definitions.- Data transformation terms used in this paper are defined below.

Standard representation (SR): A machine-independent method of encoding information when messages are sent between machines in the network

SR record: A bit string containing one or more items of information encoded in SR (may begin and end on any bit boundary)

Machine representation (MR): The machine- and compiler-dependent encoding of information; differs for each machine architecture and compiler

MR record: Group of information encoded in MR. Each item in an MR record is on an appropriate boundary for that type of item on that machine. MR records begin and end on machine-addressable boundaries.

Record descriptor: A specification of the contents of a record. For a given sequence of values, the record descriptor uniquely determines both the SR record format and the MR record format for that set of values. Record descriptors are used to guide translation between MR records and SR records.

Field mask: A specification of the presence or absence of fields in an MR or SR record, permitting data transformation routines to extract a set of fields from, or insert a set of fields into, a record

SR routines: A set of routines (subset of IPSR) for translating between MR and SR, grouped under the following headings:

Record translation routines
Item translation routines

SR map: A specification of the beginning bit positions and bit lengths of the fields in an SR record

MR map: A specification of the beginning word position and length in words of the fields in an MR record

Module Specification.- Since MRs are both machine- and compiler-dependent, most of the modules below exist in multiple versions. The names of the modules include a letter to indicate (1) which MR the routines are designed for and (2) which language they are designed to be called from. This position in the module name is indicated by a small x in the following documentation. The actual replacement for this x would be:

x	Calling Language	MR
P	PASCAL	PASCAL
F	FORTRAN	FORTRAN
X	PASCAL	FORTRAN

The module names are given below with their descriptive titles. These routines require the bit manipulation routines.

Record-oriented routines:

ESZxMX translate SR record into MR record

ESZxSX translate MR record into SR record

ESZxSM produce SR map from record descriptor and field mask

ESZxMM produce MR map from record descriptor and field mask

Item-oriented routines:

ESZxSI Translate integer value from MR to SR

ESZxSR Translate real value from MR to SR

ESZxSC Translate characters from MR to SR

ESZxSL Translate logical values from MR to SR

ESZxMI Translate integer value from SR to MR

ESZxMR Translate real value from SR to MR

ESZxMC Translate characters from SR to MR

ESZxML Translate logical values from SR to MR

ESZxSD Provide bit length of SR for a specified type

ESZxMD Provide alignment and size for specified MR type

Record-oriented routines (module description):

ESZxMX: Translate from standard record to machine record

Inputs: standard record
record descriptor
field mask for standard record
field mask for machine record

Outputs: machine record
length of standard record processed (in bits)
return code

ESZxSX: Translate from machine record to standard record

Inputs: machine record
record descriptor
field mask for machine record
field mask for standard record

Outputs: standard record
length of standard record produced (in bits)
return code

ESZxMM: Produce an MR map

Inputs: record descriptor
field mask for machine record
mode flag

Outputs: MR map (if mode flag=0)
length of machine record (in memory units)

ESZxSM: Produce an SR map

Inputs: record descriptor
field mask for standard record
mode flag

Outputs: SR map (if mode flag=0)
length of standard record (in bits)

Item-oriented routines (module description):

ESZxSI: Translate integer from MR to SR

Inputs: bits of precision in MR value
integer in MR
bits of precision in SR value

Outputs: integer in SR
length of SR (in bits)
return code

ESZxMI: Translate integer from SR to MR

Inputs: bits of precision in MR value
integer in SR
bits of precision in SR value

Outputs: integer in SR
length of SR (in bits)
return code

ESZxSR: Translate real from MR to SR

Inputs: MR precision (in bits)
real in MR
SR exponent precision (in bits)

SR fraction precision (in bits)

Outputs: real in SR
length of SR (in bits)
return code

ESZxMR: Translate real from SR to MR

Inputs: MR precision (in bits)
real in SR
SR exponent precision(in bits)
SR fraction precision (in bits)

Outputs: real in MR
length of SR (in bits)
return code

ESZxSC: Translate characters from MR to SR

Inputs: mode, selects from possible machine representations
characters in MR
number of characters

Outputs: characters in SR
length of SR (in bits)
return code

ESZPxMC: Translate characters from SR to MR

Inputs: mode, selects from possible machine representation
characters in SR
number of characters

Outputs: characters in MR
length of SR (in bits)
return code

ESZPxSL: Translate logical values from MR to SR

Inputs: mode, selects from possible machine representations
logical values in MR
number of logical values

Outputs: logical values in SR
length of SR (in bits)
return code

ESZPxML: Translate logical values from SR to MR

Inputs: Mode, selects from possible machine representations
logical values in SR
number of logical values

Outputs: logical values in MR
length of SR (in bits)
return code

ESZxMD: Provide information about MR for item

Inputs: type of item
modifier, selects from possible machine representations
n, number of values for character or logical types

Transput: index into record (ESZxMD aligns it)

Output: size of MR (in memory units)

ESZxSD: Provide information about SR for item

Input: type of item
P1,P2 = parameters for type

Outputs: size of SR (in bits)

INSTRUMENTATION AND PERFORMANCE MEASUREMENT

Overview

A performance data-gathering facility is provided so that IPAD and the host computer it runs on can be instrumented to collect performance data. When reduced, this performance data can be used to answer critical performance questions such as:

What is IPAD's response time or, alternatively, how fast does IPAD run on this particular host computer system?

Why does IPAD execute the way it does on this host?

What has to be done to change the way IPAD executes?

A two-part data-gathering facility is provided. One part collects IPAD-related performance data that is used to:

1. Provide program diagnostic trace information
2. Identify component usage and system bottlenecks
3. Optimize the internal design of IPAD components
4. Relate a component's use of host system resources to its operation
5. Provide response times

The other part of the data-gathering facility operates independently to collect host system execution data that provides a measure of the IPAD host system's ability to process the IPAD workload.

These data-gathering elements generate streams of performance measurement data, which are recorded for later analysis.

Finally, collected IPAD performance data containing execution traces, component internal data, host system resource usage data, and host system execution data are reduced and performance reports are generated. Reduction can be performed on any host.

IPAD's Use of Data Gathering

IPAD uses the data-gathering facility (IDGS) to enable IPAD and application programs to determine their resource usage and performance. These uses are described in the following subsections.

Evaluation of System Response and Performance.- IPAD designers use the IDGS trace facility as a diagnostic tool to track a transaction's progress through IPAD. This feature is particularly useful when multiple transactions are being processed by the IPAD relational data base system. After separate processing, a printed record of each transaction's progress through the various IPAD components is produced. If the recording of time has been requested, the amount of time a transaction spends in each component can be determined. This enables processing bottlenecks to be identified so that remedial development and optimization actions can be taken. Also, the effect of a particular module's design on IPAD's overall performance can be evaluated.

A component's trace and resource usage data can be allocated to itself and its designated internal blocks through the technique of bracketing. By this means measurement data can be grouped into a hierarchy of levels whereby the stream of low-level measurements is assigned to designated components and component blocks of the IPAD task. With this information, higher-level events can be constructed from lower-level events by the performance data analyzer.

The amount of data collected at each level may differ, permitting the capture of only the most useful data and with only the level of detail required.

IPAD Structure Optimization.- Component frequency-of-use statistics can be developed from the trace data. These statistics provide component locality maps for analyzing the structure of

overlay trees in those IPAD hosts that do not have automatically paged (virtual) memories.

Component Optimization.- IPAD components can collect internal data for the development and optimization of component performance.

The resource manager function of the IPAD data manager can be used as an example of the kind of task-specific data that might be collected. Components of the data manager requiring file access will use the record manager, which, in turn, will use the IPAD file control system. A representative action might be to satisfy a request to add a data record to a named file. To do this, the record manager examines a table resident in main storage to obtain the file address of a physical block containing sufficient free space for the new record. The record manager then calls the buffer manager to access the block.

Data Needed: For this representative action to be optimized, the following kinds of data are needed.

1. Disk access: The number of disk accesses needed to satisfy a request for a given-size block

2. Size of requested blocks: The size of the blocks requested by the various components of the IPAD data manager

3. Free space available: A record of the amount of free space available in the accessed blocks of the file

4. Locking interval: The length of time a working storage buffer or file buffer remains locked

Data Analysis: This kind of component internal data can be used to optimize component parameters such as:

1. The relationship between the size of the record to be inserted in a file block and the extent of the free space range. (A distribution of record sizes will enable the boundaries of the free-space ranges to be chosen so as to enclose naturally occurring clusters of record sizes.)

2. The size of the record to be inserted in a file's block versus the amount of free space remaining. (This will provide an indication of the fragmentation of available free space.)

3. An evaluation of the number of free-space chains versus the number of blocks accessed (i.e., the extent of relinking needed).

Data Recording: Task-specific data need not be recorded as it is generated. Composite records of timing, storage and event data can be accumulated until the end of the component's processing. An example of this kind of data is the IPAD data manager activation record, which will be generated by the IPAD data manager scheduler component.

Resource Characterization of IPAD Jobs.- IPAD components can capture their use of host system resources. With this information, IPAD designers can associate transaction complexity with resource usage and create data to be used for component and system optimization. More directly, a designer can see which components act as bottlenecks in restricting system flow.

Aggregate consumption of host system resources represents the task's accumulated use of the host system; it is combined with resource usage data from many other IPAD jobs to develop a concise description of the IPAD job mix. This characterization of IPAD jobs permits the identification of the resources needed by IPAD on a host system.

Evaluation of a Host as an IPAD Processor.- The IPAD data-gathering facility captures host system data that, when analyzed, is used to establish performance of an IPAD host computer as a processor of the IPAD workload. The host's ability to process IPAD jobs establishes its throughput rate and IPAD responsiveness to user actions. A host's job processing ability is governed by both its hardware characteristics (CPU speed, I/O channel capacity, and device access times) and its operating system software (maximum multiprogramming level, job priority assigned by class, etc.).

The performance data collected will be used in the IPAD performance model, a queued network of servers representing components of the host system (CPU, I/O devices, etc.) and their associated queues. The ability of these components to process work is their service rate, which is derived from the measured host system statistics (host capability data).

Collection of IPAD Performance Data

Data Gathering Design and Function.- Due to its need for computing power, IPAD will operate on multiple, interconnected, and possibly different host computers. This means that IPAD software should be as host-independent as possible. For the IDGS function, host independence is obtained through use of a two-layer interface: a host-independent part, with which IPAD components communicate, and a host-dependent part, which communicates with the host and is tailored to the characteristics of each host.

IPAD performance measurements are collected with a two-part data-gathering facility. The first part is a passive subroutine attached to the IPAD task, components of which control their own data gathering and collect performance data through calls to this data-gathering subroutine. They are provided with the ability to (1) start and terminate the IDGS gathering operation; (2) globally set the amount of detailed data to be gathered; (3) enable blocks of code to be bracketed for hierarchical data grouping; (4) enable local setting of data volumes; (5) within a bracketed code area, record occurrence of designated events; and (6) record a component's self-generated internal data.

The second part operates independently of IPAD to collect execution and resource data for all jobs processed by the host system. This job is performed by a highly host-dependent data gatherer, which periodically samples key host system performance variables.

Types of Data Collected.- The performance data collected from IPAD components (as distinguished from host system data) fall into three types. The first is trace data, which records the internal execution of an IPAD component. The second type is component-specific data, which is generated internally by the IPAD component. Its format and meaning are specified by the component and can represent loop counts, storage used, and timing data. The third type represents actual usage of the resources of the host system, which include CPU time, number of I/O accesses, and amount of main storage occupied.

Format of Collected Data.- Each measurement generates a specific group of data elements. Each group is of varying length, consisting of an identifying header and a variable-length data component, depending on the event and level of detail requested. Each group is appropriately identified by IDGS and, depending on the level of detail requested, may contain event time and CPU time. If generated by a host interface routine, detailed host data will be present in addition.

The IPAD performance data measurements are purposely generated in a host-independent format to ensure that the reduction function on one host can process the performance measurements collected by another host of different characteristics. This is further ensured through the use of a standard data format and data type.

Collection of Trace Data.- An IPAD component's trace data is generated by the component and its host interface routines, as they execute, through subroutine calls to IDGS. The amount of

trace data generated by each component can be controlled by the component, depending on the expected use of the data.

Four levels of data-gathering detail are provided. At the first level, the trace contains only a record of the calls to the data-gathering facility, including both the implicit trace generated by the host interface routines and the explicit trace generated by the event and bracket commands. At the second level, each trace event is time stamped (i.e., the current value of wall clock time is added to the trace's measurement group). At the third level the accumulated CPU time consumed to that point by each IPAD task (accumulated by the host system and accessed through host system services) is added to the measurement group. At the fourth and final level, special-purpose detailed data is added. This data, consisting of time stamps, I/O device identification, and physical record counts, is obtained only from the specially modified IPAD development host's operating system.

Collection of Frequency-of-Use Statistics.- Component frequency-of-use statistics can be gathered by instrumenting each component or component block of interest with a call to IDGS to create an event denoting its execution.

Collection of a Component's Internal Data.- To collect this type of data, IPAD component designers examine the function or service of their component and identify those features that characterize or govern its performance. These features probably will be unique for each component.

The kinds of information to be gathered to characterize the operation and performance of the component are typically:

Timing: Timing data can be captured through the recording of events such as termination of a major loop or how long another component takes to provide a service for this component. An IDGS function will supply wall clock time.

Storage: Recording queue lengths and the size of working or I/O buffers facilitates determination of the relationship between a record's size and the sizes of available free space, or the amount of free space available for allocation. The data-gathering facility can measure the amount of storage occupied by the IPAD component only when it is executing, and then only in terms of the component's data structures. It can do this by recording the size, units, and numbers of instances of these structures.

Activity counts: These can be simple recordings of component-maintained counts of various kinds of events that are meaningful to the task. For example, how many elements does a queue have now? How many calls to the resource manager have to be made to

access a B-tree page? What is the number of scheduler cycles needed to process a request?

When parameters specifying the internal performance of the component have been identified, the designer's next step is to specify numerical values and identification codes. These timing, storage, and activity measurements are accumulated by the component in an array, with a format determined arbitrarily by the component's designer, and then output through subroutine calls to IDGS.

Collection of Host Resource Data.- Access to host resource usage data is accomplished primarily through host system services. These services usually supply accounting-type data of the host resources consumed by each host, whether IPAD or other. An IPAD job's use of host system resources, independent of those collected by the host system accounting functions, is collected by the data-gathering facility, either during its operation (through tracing, as previously mentioned) or at the end of its execution. The resource use measurements are placed at the end of the task's data stream.

Collection of Host System Measurements.- Host system processing capability data will be collected by a data-gathering facility that is hooked into the host computer's operating system. It collects the following types of key host operating system information (expressed in CYBER host terminology):

1. Number of concurrent users by job class (number of control points executing IPAD jobs)

2. CPU and I/O busy times (measurements of the "busyness" of the CPU and I/O channels of the host system)

3. Number of completed jobs

4. Number of completed operations per device (total number of reads and writes transferring physical blocks of data)

5. Total number of I/O operations (number of I/O operations for each device calculated for all I/O devices, including terminals)

6. Measurement interval (time interval over which the measurements are made)

7. Job execution status (breakdown of the execution status of the jobs being multiprogrammed in the system)

Data Reduction Report.- The IPAD performance report first presents the IPAD task's resource usage in the form of a matrix, with resource types (e.g., CPU time) as columns and component or block identifiers as rows. Another column shows the level of data gathering specified for each block. These and the block listings will show the hierarchy specified by the instrumentation bracketing. If the IPAD task was not instrumented for resource measurements by individual components, there will be a single row showing only the task's accumulated use of host system resources.

The second section presented in the trace report details events (i.e., designer-defined happenings in a component, recorded by a call to the IDGS subroutine). Events are listed by identifier and ordered by time of their occurrence.

The third section of the trace report lists the measurements generated internally by the IPAD task or its component parts. These measurments are peculiar to the program generating them, except for their standard but variable-length format of length, identifier, and the data. In the absence of a decoding template, such as a data structure, a default listing of one recording instance per row will be generated. Rows will be ordered by the time of their occurrence.

Reduction of Trace-Type Data.- Trace-type measurements are generated by the execution of the various components of each IPAD task. These measurements appear as a string of identified groups, each consisting of length, identifier, and data, which are scanned to extract basic performance information for presentation in the report.

Reduction of Frequency-of-Use Data.- Component frequency-of-use data are reduced to generate overlay tree map data, each branch of which is identified along with its use count.

Reduction of Component Internal Measurements.- The performance measurements generated internally by an IPAD component have meaning only to that component. The content of the measurement (timing, storage, and counts) is generated dynamically by a component's operation and depends on the flow of execution. These measurements have a common format; their presentation in an IPAD performance report will require special-purpose processing defined by the component's designer.

Reduction of Job Characteristic Measurements.- IPAD usage statistics will be reduced to produce measures of the host system

resources used by each IPAD job, such as elapsed time, CPU time, amount of memory occupied, number of I/O accesses made, and number of files accessed.

Processing will involve cluster analysis and scaling techniques, in which the collected resource information is grouped and averaged to produce repesentative job descriptions. In addition, reduction produces job resource usage inputs for the IPAD queued network performance model. These inputs consist of elapsed time, CPU time, I/O activity, and storage requirements for each job.

Reduction of Host Characteristic Measurements.- The following kinds of performance model parametric data are produced from collected host-specific data:

Multiprogramming level

CPU service time by job class

I/O device service time by job class

Host system component routing frequencies

Job swapping activity

Job residency times

Summary of IPAD Performance Measurement and Reduction

The intent of the data gathering part of this paper has been to provide an operationally oriented, comprehensive description of the total IPAD IDGS facility. The presentation was mentioned in three major sections: performance data use, performance data collection, and performance data reduction.

IPAD's use of IDGS was considered first. This showed how the various types of measurement data were used to measure IPAD's actual performance and use of host computer resources.

Described next were (1) a description of the IDGS software, (2) the types of performance measurements gathered, and (3) the mechanisms by which the measurements were collected.

The last section briefly described the performance report format and outlined the processing of the measurement data.

SYSTEM DEVELOPMENT USING A HIGH-ORDER LANGUAGE (HOL)

More and more, high-order languages (HOLs) are being used to design and build programming systems both large and small. Efficient use of CPU and memory resources when building programming systems, although important, is now less critical because of more efficient object code output by compilers, the developments in increased machine speed, more sophisticated machine architecture, and larger and cheaper memories. It makes good sense to choose a programming language best suited to solve the problem at hand. For example, most programmers would agree that numerical algorithms are best represented in an HOL such as FORTRAN and not in assembly language or an HOL like SNOBOL and especially not in assembly language. The same arguments for HOL use in applications are being used to justify the use of HOLs in designing and building system software.

Pascal as the HOL

In the early stages of the project, a study was performed to select a programming language for implementing the IPAD system (ref. 6). From an initial list of 15 languages, five were chosen for detailed evaluation. These were AED, FORTRAN, LITTLE, Pascal, and Assembly Language with Macros.

The languages were evaluated on 20 different weighted criteria, among which were degree of standardization, portability, object code efficiency, support of complex data structures, and support of structured design and programming. As a result, Pascal was selected as the system design and implementation language. Our discussion of Pascal's advantages and disadvantages is in the context of its use in development of operating system interface and network software.

Advantages

The advantages Pascal affords the designer and developer of system software are those that it provides any programmer. Some of those advantages are:

Very well structured control statements

Supports a wide variety of data structures

Recursive

Extensive type checking

Portability

Optional run time error checking

The structured control statements, wide variety of data structures, and extensive type checking have aided in the top-down, fully specified approach we have taken with our software design and devlopment. The data structures and type checking have forced us to think through and precisely define our data structures and module interfaces. These language features, along with our software standards (ref. 7), have resulted in well documented and maintainable software. The optional run time error checking allowed us to thoroughly check out our programs and then to turn off checking in production code releases for better performance.

Disadvantages

Some of our problems with Pascal are general, others are perhaps unique to the development of system software. Some of the problems we encountered were:

Weak I/O facilities

Type checking sometimes too restrictive

Variable-length arrays not generally supported

No explicit data initialization statements

Too much run time environment required

No "own" type variables (as in ALGOL)

Heap allocation not optimal

When developing a general set of service routines, particularly I/O routines, it is not possible to predict all of the forms that the data in the I/O buffer will take. Length is also an unpredictable variable for a data buffer. Type checking and the lack of variable-length (conformable) arrays forces the systems programmer to compile the service routines externally. Each applications programmer must then define his own procedure declaration with the proper buffer type. These buffer types will generally differ, which seems inconsistent with the intent of Pascal's type-checking mechanism. Such problems are implementation-dependent. One difficulty is the lack of Pascal standards for external compilations and for modularizing large systems. In fact, if such a standard existed and were universally or generally implemented, the "tricks" employed to avoid type checking would no longer work. This should happen, and when it does a more comprehensive solution to the "unknown data type" problem will be necessary.

Compiler implementations are also inconsistent in their type checking. Some require structural equivalence; others require name equivalence; still others use declaration equivalence, which is slightly less stringent than name equivalence. (Our compiler enforces declaration equivalence.) The programmer is generally able to initialize variables at compile time but only global variables in the main program. Another very useful compiler feature that is generally lacking is the use of expressions in defining constants. An absolute requirement, which is provided by most compilers, is the support of packed data structures.

The Pascal run time environment assumes that a procedure or function is being called from the context of another Pascal program. This assumption makes it difficult to use Pascal to define interrupt handling procedures or any procedures which execute asynchronously. In building general-service software, it is useful if not mandatory to have a way to define static variables that are not part of the stack structure and therefore remain defined between calls.

General Experience

Coming from many years of experience in designing, building, and maintaining system software using assembly language and macros on a variety of machines, some assembly-language-oriented system programmers had a difficult time making the transition to Pascal, where accessing a bit in a byte or word for the first time proved to be a frustrating and agonizing process. An example of accessing fields in a system control word using Pascal is shown in figures 11 and 12. One of the major benefits has been productivity. In a relatively short time (6 to 12 months), a group of three or four people has developed executive and network software that is easier to debug and maintain than its assembly language equivalent. Ease of maintenance is an important benefit; maintenance runs from 70 to 85 per cent of the cost of software over its life cycle.

On balance, our experience with Pascal has been a good one. Its use has forced a more disciplined and rigorous approach to the solution of problems. Assembly language still has its place in the development of software that must inteface with the operating system and machine hardware, but it is hard to imagine returning to assembly language as a vehicle for building general system software.

CONCLUDING REMARKS

Complete host system independence is a desirable goal for IPAD software components. We have achieved this independence to a high degree, but at the expense of not using or granting access to useful host system services that are unique to some systems. This method of isolating IPAD software components from the host system has minimized host system dependent code. What dependencies exist are localized in a small number of modules.

Writing executive and communication software in a high-order language has been challenging and, generally, successful. We have experienced high productivity and have been able to concentrate on the important problems of design and implementation. Most of the problems we encountered were not with the Pascal language but with the implementation of the compiler and the run time subsystem. Lack of computer vendor support for one of our Pascal implementations was a problem. More vendor support may have meant a compiler and run time subsystem that was better integrated with system architecture and system services.

REFERENCES

1. IPAD Computing Staff: Level II IPAD, IPAD document D6-IPAD-70036-D-9, March 1980.

2. Anderton, G. L, et al: IPAD User Requirements, IPAD document D6-IPAD-70013D, August 1977.

3. Meyer, D. D., and Southall, J. W.: First-Level IPAD User Requirements, IPAD document D6-IPAD-70016-D-1, March 1980.

4. Southall, John W.: Requirements for Company-Wide Management of Engineering Information. IPAD: Integrated Programs for Aerospac Vehicle Design, NASA CP-2143, 1980. (Paper 7 of this compilation

5. Ives, F. M., et al: IPEX Preliminary Design, IPAD document D6-IPAD-70036-D-4, March 1980.

6. IPAD Computing Staff: IPAD Evaluations and Alternatives, IPAD document D6-IPAD-70036-D-3, March 1980.

7. Warkentine, E. R.: IPAD Software Standards Handbook, IPAD document D6-IPAD-70014-D, April 1979.

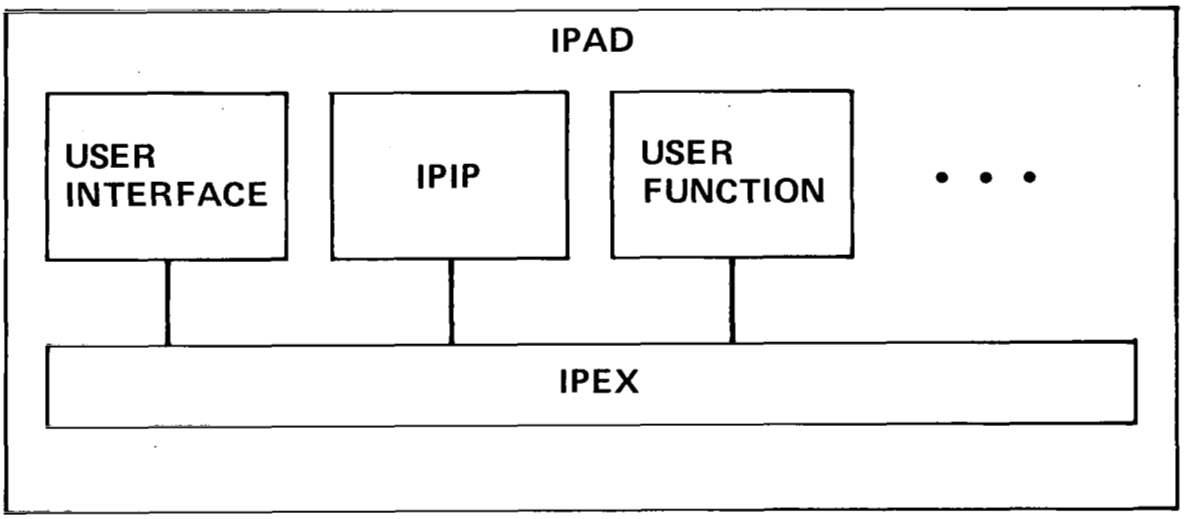


Figure 1.- Logical relationship of IPAD components.

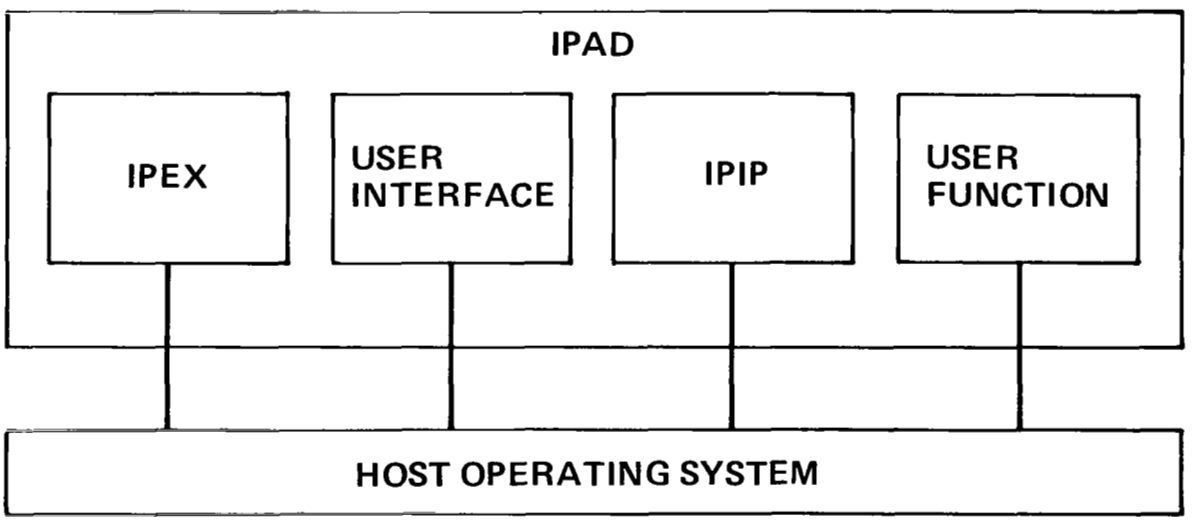


Figure 2.- Physical relationship of IPAD components.

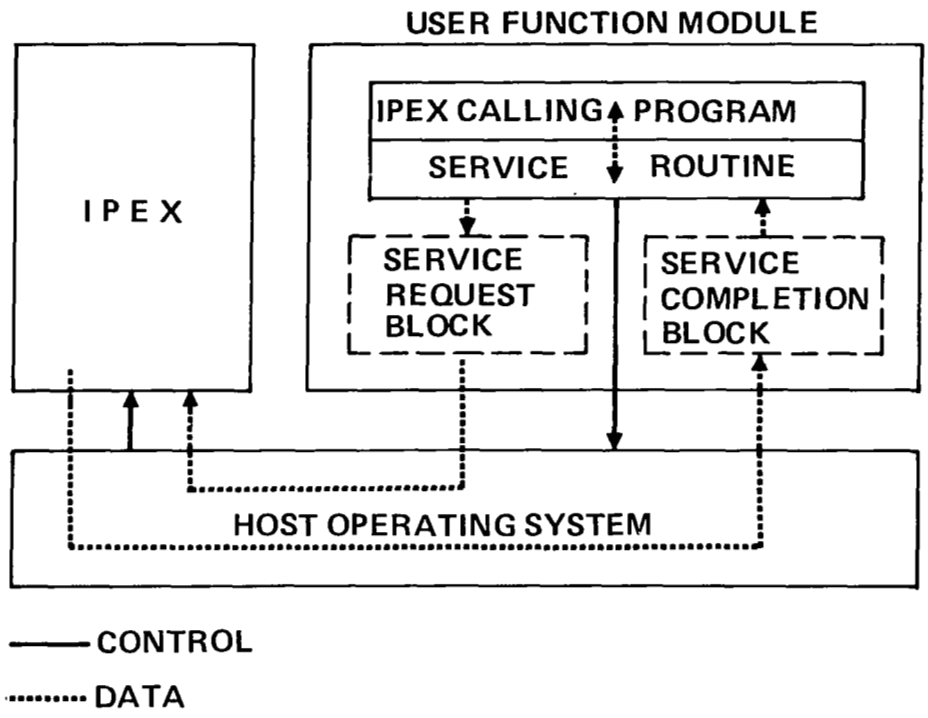


Figure 3.- IPEX to task interface mechanism.

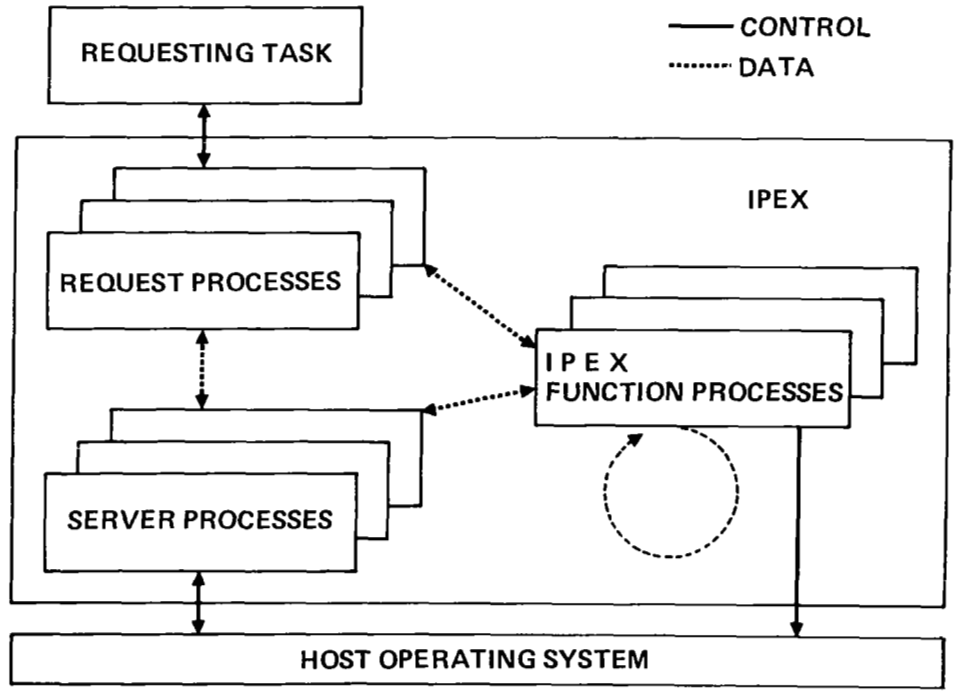


Figure 4.- IPEX internal design model.

	PARAMETERS											
FUNCTIONS	FILNAM	BUFAD	BUFLEN	ACCESS	SHARE	PASSWRD	SYNCH	STATUS	INDEX	ACTION	CREATE	RECSIZ
• OPEN	X			X	X	X	X	X			X	X
• CLOSE	X						X	X		X		
• TRUNCATE	X						X	X	X			
• DELETE	X					X	X	X				
• READ	X	X	X				X	X	X			
• WRITE	X	X	X				X	X	X			
• POSITION	X						X	X	X	X		
• WAIT	X							X				
• CHECK	X							X				

Figure 5.- File I/O functions.

	PARAMETERS									
FUNCTIONS	NAME	PORT#	UNITS	LENGTH	MESSAGE	MODE	ID	TIME	CLASS	STATUS
• CONNECT	X								X	X
• DISCONNECT										X
• SEND	X	X	X	X	X	X	X	X		X
• RECEIVE	X	X	X	X	X	X	X	X		X
• CHECK							X			X
• WAIT							X	X		X
• CANCEL							X			X

Figure 6.- Network I/O functions.

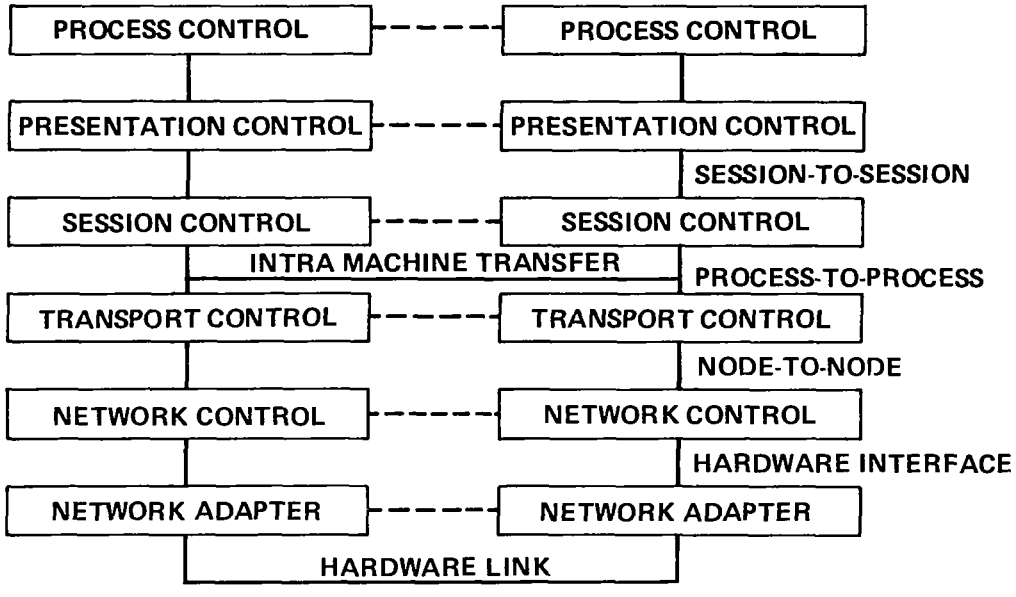
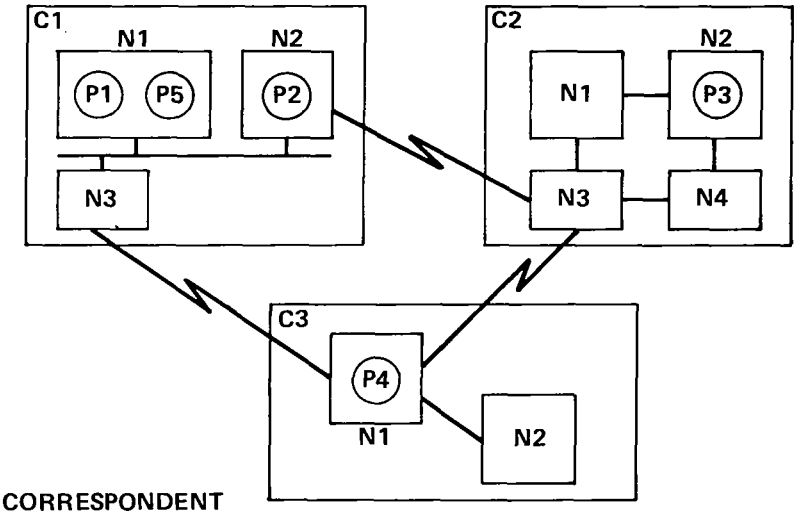


Figure 7.- Logical layers of communication.



CORRESPONDENT PROCESSES		SOURCE ADR	DESTINATION ADR
FROM	TO	P1	P5
P1	P5	N1 . P1	N2 . P2
P1	P2	C1 . N1 . P1	C3 . N1 . P4
P1	P4	C1 . N2 . P2	C2 . N2 . P3
P2	P3		

Figure 8.- Addressing in a three-cluster network.

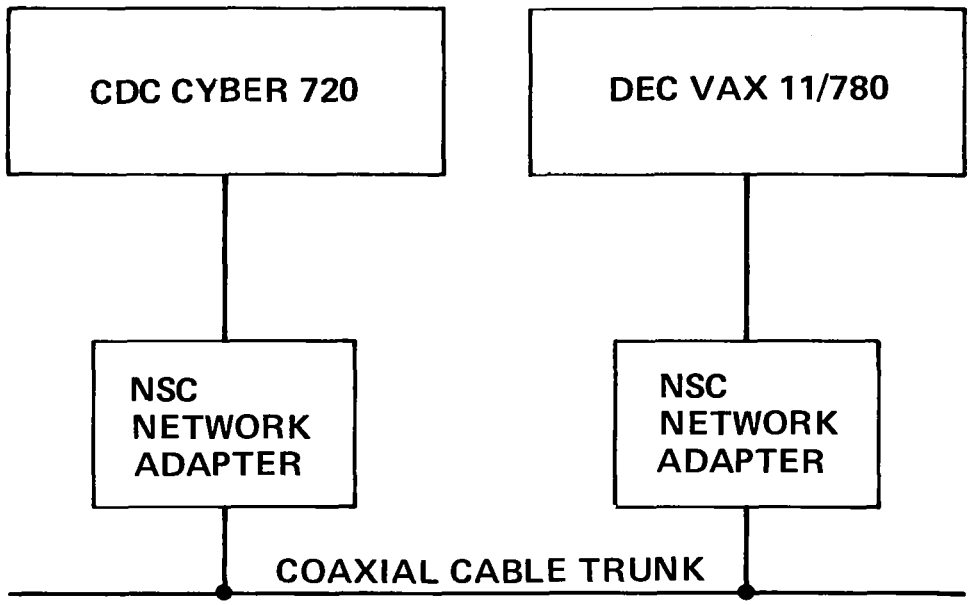


Figure 9.- Local network configuration.

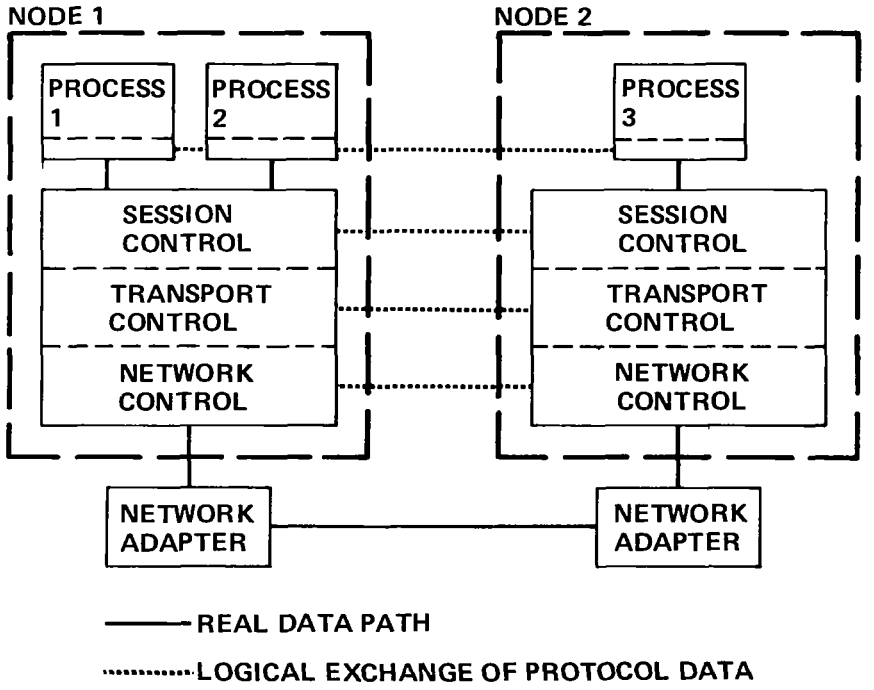


Figure 10.- Communication system components.

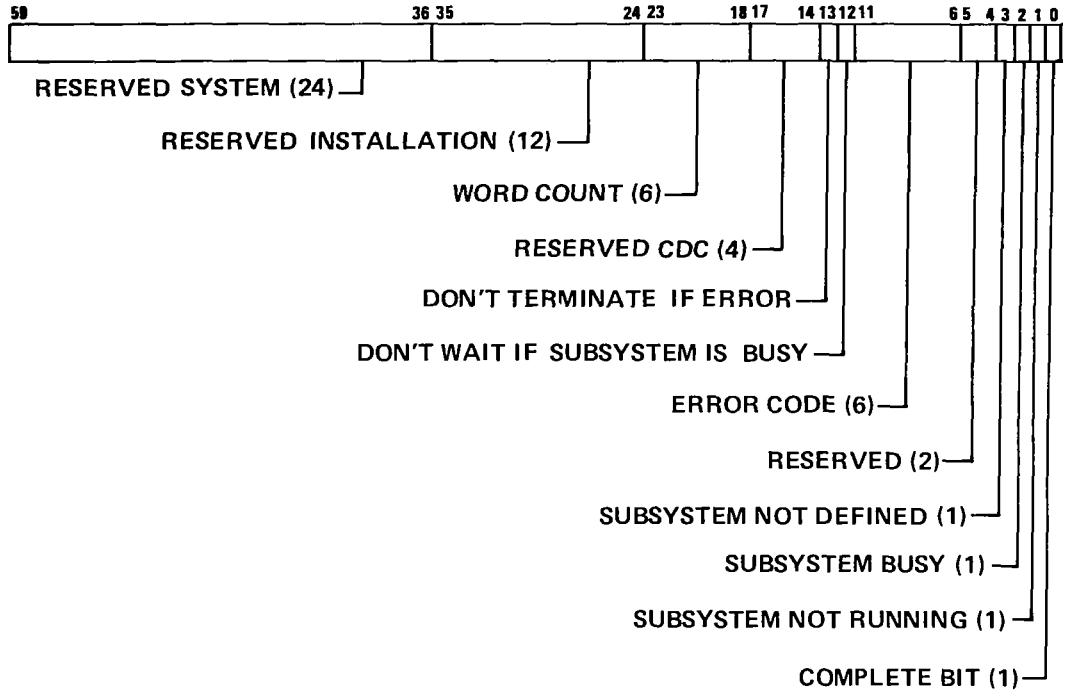


Figure 11.- Typical system control word format.

```
BIT = 0 . . 1;

ETTCWD = PACKED RECORD
           RSVDSYS      : PACKED ARRAY [1 . . 24] OF BIT ;
           RSVDINST     : PACKED ARRAY [1 . . 12] OF BIT ;
           WORDCNT      : 0 . . 7 B                      ;
           RSVDCDC      : PACKED ARRAY [1 . . 4]  OF BIT ;
           NOTERM       : BOOLEAN                        ;
           NOWAIT       : BOOLEAN                        ;
           ERRSTAT      : PACKED RECORD                  ;
             ERROR      : 0 . . 77B                      ;
             RSVD       : PACKED ARRAY [1 . . 2]  OF BIT ;
             NOTDEFND   : BOOLEAN                        ;
             BUSY       : BOOLEAN                        ;
             NOTRUN     : BOOLEAN                        ;
             END                                         ;
           COMPLETE     : BOOLEAN                        ;
           END                                           ;
```

Figure 12.- Pascal description of a word.

AN ENGINEERING DATA MANAGEMENT SYSTEM FOR IPAD

H. R. Johnson, D. L. Comfort, D. D. Shull
Boeing Computer Services Company

SUMMARY

This paper presents an overview of the capabilities and software architecture of the IPAD information processor (IPIP). IPIP is a state-of-the-art data base management system that satisfies engineering requirements not addressed by present day commercial systems. It also significantly advances a number of capabilities that are offered commercially. IPIP capabilities range from support for multiple schemas and data models to support for distributed processing, configuration control, and data inventory management.

IPIP exploits semantic commonality in features offered in various forms at different user interfaces in today's commercial systems. An integrated software architecture supports all user interfaces: programming languages, interactive data manipulation, and schema languages. This approach promotes simplicity and compactness in software and permits features to be offered symmetrically across all appropriate user interfaces. Thus, more functionality is provided for the user in a more uniform and usable fashion.

DATA MANAGEMENT REQUIREMENTS

IPIP was originally envisioned as a data manager to support the design process for aerospace vehicles. Consequently, an experienced team of Boeing engineers assigned to the IPAD program conducted industry-wide surveys and interviews to determine engineering design requirements. These are documented in references 1 and 2.

Since that time, the importance of data sharing and continuous flow of data between the engineering and manufacturing processes has come to be more fully recognized. The conclusion has been that IPIP should support both computer-aided design and computer-aided manufacturing. This requirement was reflected in a resolution passed by attendees at the the ICAM/COCAM Data Base Workshop in Dallas in April 1979 to the effect that IPIP should be used to support ICAM data base management. Currently, an IPAD team is working with Boeing manufacturing to assess IPIP requirements in relation to manufacturing requirements.

Engineering Requirements for Data Base Management

The DBMS requirements discussed in this section were derived from the IPAD system requirements specified by the IPAD engineering team. These were identified through the use of both industry-wide surveys and on-site interviews with representatives from the U.S. aerospace industry. This effort resulted in the formulation of the following categories of DBMS requirements:

1. Multiple views of data
2. Multiple levels of data description
3. Dynamic data definition
4. Distributed data (base) processing
5. Extendible data types
6. Geometry processing
7. Configuration control
8. Data inventory management

Each of these categories will be discussed in some detail below.

Multiple Views of Data.- A characteristic of the data bases found in this environment is that they are typically on the order of billions of bits in size. Because of their large size and inherent complexity, users will process subsets of the data base at one time (i.e., no one user will have the need to process the entire data base at any given instant); this requires the DBMS to support the definition and manipulation of subsets (e.g., sub-schemas or logical schemas) of a more global view of the data base. Furthermore, one finds upon investigation of the type of data processed that there is a need and desire to organize the data into logical structures such as hierarchies and networks; however, a large portion of the data used in the design process lends itself to being organized into tables or relations. Thus we see the need for not only supporting the viewing of portions of a data base but for organizing these portions into different formats (hierarchies, networks, relations) of data. Stated more explicitly, IPIP must be able not only to support subsetting of the data base into comprehensible partitions for users, but also to do this in the form of multiple data models. The data models supported by IPIP are discussed in a later section of this article. These different views of data may be processed by FORTRAN application programs, Pascal application programs, ad hoc interactive users, and complex graphics software.

Multiple Levels of Data Description.- The ability to localize the impact of change to data definition is referred to as data independence. To provide the data independence required to accommodate the aerospace design environment, multiple-level arrangements of schemas must be supported in such a way that the impact of change to any one schema is insulated with respect to other schemas and applications. IPIP supports a schema-application environment (or data architecture) that permits very flexible configurations of schemas and applications. The IPIP data architecture provides for muliple levels of schemas. To enhance performance in this multilevel data architecture and minimize the need for recompilation, IPIP supports facilities for binding applications to data definitions at various times, independent of the compilation process.

Dynamic Data Definition.- Preliminary investigations into the record types and relationships needed to support the aerospace design environment have shown that it is impossible to develop a complete, correct definition of the total data base during the initial phase of schema development. The size and complexity of the data dictate that the development of the schemas must be an evolutionary process. Data administrators of IPIP data bases will need to be able to dynamically modify the data base description without having to recompile existing schemas and application programs (except under circumstances where definitions are removed from the schemas); i.e., it must be possible to enhance schemas as knowledge of record types and relationships becomes known through experience.

Additionally, users may wish to define new and/or modify existing data definitions (schemas) in their programs to create "temporary" data bases comprising new record types and new data.

We categorize the dynamic modification of schemas, whether by data administrators or users, as "dynamic data definition."

Distributed Data Processing.- All types and sizes of computer hardware are used in a typical aerospace engineering firm. With the increasing demand for the sharing of data between different computers within a firm, we find computer networks involving heterogeneous computers. In these configurations there are collections of both large-scale mainframes and minicomputers, and there is a need to share data between them. These machines may be connected via local high-speed communication links (50 megabits/sec) and/or global low-speed links (300-9600 bits/sec). Figure 1 depicts a sample hardware configuration for a distributed environment.

A user residing at sites A, B, and C will process his own data locally and require local data management services but will

also need to access data on different computers at different sites using data management capabilities. Therefore, there must be IPIP functionality not only at a single site but at multiple sites as well. Ideally, users will not need to know the physical location of the data, so the DBMS must be able to determine the site at which data resides or is to be stored.

In order to provide IPIP data management functionality at multiple sites, IPIP must be resident at some sites and accessible to all sites. How multiple IPIPs would communicate with each other is beyond the scope of this paper.

Figure 2 shows a sample configuration of multiple sites with multiple IPIPs available to users. The residency of an IPIP DBMS at a site is not a prerequisite for the availability of data management services at that site. Users at site A could use data management services at site B through IPAD network services.

Extendible Data Types.- IPIP must accommodate definition of new and different data types (IPIP must be extendible) as needs arise.

Geometry Processing.- Amid the complex data bases in the engineering world is found a characteristic not present in business data management: the need to define and manipulate geometric entities. This geometric capability places special requirements on the data management system. The system must manage geometry, enforce constraints, maintain continuity, and track associations of entities. It must also allow the manipulation of geometry data at the entity level. These requirements have resulted in some unique facilities in IPIP.

Configuration Control.- During the design of an airplane, hundreds of engineers will create (and store in a data base) hundreds of versions of wings, fuselages, engines, etc. There is an administrative need for each engineer to maintain control over his data during a given phase of the design process, and this translates into a need to permit only the responsible engineer to have modification rights until the design of the aircraft component has reached a certain phase. At this point, a particular design is designated as being officially released, and no further updates may occur thereafter on this data; however, when modifications of the official version are required, these changes will result in the generation of new versions of the existing data. IPIP must control access to the data during both phases of design, and be able to process the data base (and portions thereof) by versions. The configuration control of the design data base is under absolute control of the engineering data base management system. Associated with each engineer's data must be a "header" describing the source and quality of the data. This

header will include such items as author's name, creation date, version of the data, and various other attributes. All of this identifying data must be maintained by IPIP.

Data Inventory Management.- This requirement is primarily a result of the five requirements previously discussed. It is best described as the need for IPIP to manage its own data about the user data. This system data (or "metadata," as it is sometimes called) consists of the following:

1. Compiled multiple schemas and the mappings between them

2. The configuration control data (headers)

3. Security information about who is processing the data base

4. A dictionary of allowable data types (e.g. points, lines, matrices, etc.)

5. Directories indicating the location of data which is distributed over a computer network

These include functions performed by data dictionaries (including commercial data directories), but additional capabilities are required to handle some of the metadata. More importantly, this inventory facility must be an active (as opposed to passive) part of IPIP in order to perform its functions correctly. Specifically, it should be an integrated part of the data management system.

Manufacturing Requirements for Data Base Management

Currently, an IPAD team is working with Boeing manufacturing to assess manufacturing requirements for data management. To date, this group has observed that, in general, engineering requirements cover manufacturing requirements, although relative emphasis on particular requirements may differ between the two environments. Multiple views and concurrent access with heavy update are very important in manufacturing data processing.

IPIP CAPABILITIES

Data Architecture

A schema is a valid sequence of statements that describe data. IPIP supports a multiple-schema data architecture that generalizes the two-schema architecture proposed by the ANSI/X3/SPARC Data Base Study Group. (See reference 3 for a

description of the ANSI data architecture.) The IPIP data architecture offers a number of advantages including the ability to vary data independence according to cost/benefit trade-offs for specific situations.

The ANSI data architecture includes three mandatory levels and types of schemas:

1. Conceptual schema--a schema that describes logical data structures for an entire enterprise, i.e., an entire data base

2. External schema--a schema that describes logical data structures derived from the conceptual schema. An external schema tailors the description of a portion of a data base to the needs of a class of applications.

3. Internal schema--a schema that describes physical data structures to support the logical data structures of a conceptual schema

Figure 3 illustrates the essentials of the ANSI three-schema architecture. The notation "m:n" in this figure indicates that m of the upper applications or schemas may be associated with n of the lower schemas. Figure 4 illustrates the hub-and-spoke arrangement of schemas and applications in the ANSI environment. The term application here refers to either an application program or an interactive (or query) session that includes data manipulation commands against a schema and hence against the data base.

In the ANSI environment, one or more internal and external schemas may be associated with a single conceptual schema. An internal or external schema is associated with exactly one conceptual schema. Interschema mapping statements are contained in internal and external schemas. An application references exactly one external schema.

The IPIP data architecture includes a variable number of levels of schemas of three types:

1. Logical schema--a schema that describes logical data structures for a collection of data

2. Internal schema--a schema that describes physical data structures to support logical data structures in a logical schema. A logical schema supported directly by an internal schema is called a base-level logical schema.

3. Mapping schema--a schema containing statements that map data structures of a logical schema onto data structures of one or more underlying logical and/or internal schemas

Figure 5 illustrates the IPIP data architecture. The notation "n:n" in this illustration indicates that n of the upper applications or schemas may be associated with one of the lower schemas. The segmented diamond represents a mapping schema that is implicitly defined by constructs in data manipulation statements within an application. One logical schema is required. The dotted loop indicates that, optionally, additional logical schemas may be included. Figure 6 illustrates some of the arrangements of applications and schemas that are possible in the IPIP environment.

In the IPIP data architecture there is only one type of logical schema instead of two as in the ANSI data architecture. Interschema mapping statements appear only in mapping schemas, not in internal or logical schemas. Applications may reference one or more underlying logical schemas at any level in the architecture. An application may not, however, reference two logical schemas, one of which is defined directly or indirectly in terms of the other. A logical schema is associated with exactly one underlying mapping schema and thereby with one or more underlying logical or internal schemas. An internal schema is associated with exactly one mapping schema and hence with exactly one logical schema.

The IPIP internal schema corresponds to the ANSI internal schema. An IPIP base-level logical schema corresponds most closely to an ANSI conceptual schema. However, unlike the conceptual schema, a base-level logical schema can be referenced by an application, and a logical schema need not define the entire data base for the enterprise.

Informally, we define data independence as the insulation of change to data definition within a data architecture. Thus, for example, in the ANSI architecture, programs and the conceptual and external schemas need not be changed when a change is made to an underlying internal schema. Programs may also be preserved in the face of certain changes to the conceptual schemas by altering mapping statements in external schemas. Generally speaking, data independence is a linear function of the number of levels of schemas used. The use of two levels of IPIP logical schemas affords the same amount of data independence as the ANSI data architecture. The use of one level of IPIP logical schema provides less data independence; the use of more than two levels of logical schemas provides more. Thus the IPIP data architecture allows the data administrator to make trade-offs between the benefits of data independence and the overhead of schema maintenance and binding by choosing the appropriate number of levels of logical schemas. The more levels of schemas, the higher this overhead. As discussed in a later section, IPIP binding options permit all binding overhead to be incurred prerun-time for precompiled applications. These options also permit most binding to be performed prior to run-time for interactive applications.

By separating mapping statements from the internal and logical schemas, the IPIP data architecture provides more data independence. For example, it is intended, in future releases, that mapping statements may span several underlying logical or internal schemas. Thus a mapping statement might choose between several underlying schemas based on the value of an item, e.g., schema S1 if the value is in range R1, schema S2 if the value is in range R2, etc. In the IPIP environment, this mapping statement may be ammended to include additional underlying schemas or to remove them without impacting existing logical or internal schemas. This is useful in distributed processing situations where nodes are to be added to or removed from the network over time.

A base-level logical schema may describe the entire data base for an enterprise as does an ANSI conceptual schema, but it does not necessarily have to. The description of the data base may be partitioned into several base-level logical schemas on a functional or geographic basis or according to the structure of a corporate organization. Thus, for example, given four departments within a manufacturing organization, there might be four base-level logical schemas describing data local to a department and one or more base-level logical schemas describing data shared by departments. This partitioning of metadata allows each department a degree of autonomy over its own metadata. This is in keeping with departmental attitudes within an organization and the growing awareness of the value of metadata as a resource.

It is not unusual for separate but related data bases to be created within a corporation. For example, these might be parallel design data bases for two or more product lines. The IPIP ability to tie logical schemas together with other logical schemas or applications facilitates the coupling of these existing data bases with minimal disruption, thereby providing more data independence. The decision to couple data bases might be an afterthought. For example, headquarters decides that it must evaluate corporate impact or relative merits of coexisting product lines. Or the decision to so couple data bases might be a strategy at the outset to provide for flexible evolution of data bases within the corporation. Reference 4 discusses the coupling of logical schemas and application of other features of an IPIP-like data architecture in a distributed data management environment.

The flexibility of the IPIP data architecture in configuring schema arrangements supports organizational practices of delegating authority. Consider the ability to define multiple levels of logical schemas. Suppose that a data administrator receives authority to read a logical schema, define logical schemas in terms of it, and delegate authority. He might then define several logical schemas based on the original, each

corresponding to the general data requirements of a sub-organization. Following that, he might delegate authority to a data administrator within each suborganization to read the appropriate logical schema and define logical schemas in terms of it. In this way, the first data administrator may concern himself with the overall data requirements of his organization and not be concerned with detailed requirements of users within the suborganizations. His subordinates may define one or more logical schemas, based on the one given them for this purpose, but they need not see the original logical schema. They need not cope with data that is not relevant to them. This simplifies their jobs and avoids potential security problems.

Subordinate administrators might be given authority to delegate authority. Thus several levels of logical schemas might be built up, depending on the structure of the organization.

IPAD requirements include provisions for delegating authority; however, the first release of IPAD will not offer authorization mechanisms.

Schema and Data Manipulation Languages

IPIP schema and data manipulation languages exhibit a high degree of integration and compatibility with anticipated standards.

There are a single logical schema language supporting multiple data models, multiple programming languages, and interactive applications; a single internal schema language sharing many constructs with the logical schema language; a single mapping schema language supporting mappings between logical schemas and between logical and internal schemas; and a single data manipulation language that may be used by both application programs and interactive applications. Thus we use the term interactive applications rather than query sessions to reflect the similarity in data manipulation capabilities in the two environments.

In some systems one might encounter one or more conceptual schema languages as well as a distinct external schema language and data manipulation language for each combination of data model and programming language supported. In addition, one or more query languages might be provided for interactive applications. By contrast, there are relatively few languages for the IPIP user to learn; for example, he need not learn one data manipulation language to access the data base interactively and another to access it via programs.

Informally, a data model is a style of defining and manipulating data. IPIP takes an integrated approach to data

modeling; i.e., a single family of schema and data manipulation languages supports both the network and relational data models or a mixture of the two. The functionality of the hierarchical data model, being subsumed by that of the two other data models, is supported by IPIP.

The logical schema language contains a DATA MODEL clause to specify which data model is being used. This clause governs which constructs may be used in the schema. For example, CODASYL sets may be declared in network schemas but not in relational schemas. Conversely, foreign keys may be declared in relational schemas but not in network schemas.

The DATA MODEL clause does not distinguish between data models for constructs that are traditionally associated with one data model but could just as well be used with the other. For example, the domain construct has historically been associated with the relational model but is a useful extension to the CODASYL network model. Thus the IPIP logical schema language permits domain declarations for either data model. Historically, interrecord operations are expressed in terms of data item comparisons in the relational data model but not in the CODASYL network data model. (This is the basis for expressing the relational join operation.) The IPIP logical schema language permits data manipulation involving item comparison regardless of which data model is specified in the logical schema.

In some systems, only one data model--say the relational model--may be used at the conceptual schema level, while two or more data models may be used at the external schema level. Support of an essentially network modeling environment in such a system would require that someone understand the relational data model and mappings between the two data models so that a relational conceptual schema could be developed. Not so in the IPIP environment, where all schemas may be network; likewise, all schemas in IPIP may be relational, or some schemas may be of one data model and some of the other, or some schemas may use constructs from both of the data models. Thus the data administrator has broad flexibiity in using data models.

Similarly, there is a host language clause in the schema language governing whether naming conventions and data item descriptions within records or relations conform to FORTRAN or Pascal syntax for declaring variables. Data item syntax of the CODASYL data definition language will be supported in future releases. Either host language may be used in any internal or logical schema. Thus with IPIP it is not necessary for someone in a FORTRAN shop to learn a second data item sublanguage in order to write a conceptual schema. Administration is simplified in the situation where applications in multiple host languages are to be supported, because a single framework is used for schemas,

irrespective of host language. Schemas differ by host language only to reflect essential differences in the host languages.

To summarize, IPIP offers a number of features that simplify the data definition/data manipulation task. This is especially important for the individual who must deal with all aspects of the process, i.e., both defining and manipulating data bases.

The IPIP logical schema language is based on a subset of the schema language being considered for standardization by the ANSI/X3/H2 committee. This language is a subset of the schema language developed by the CODASYL Data Description Language Committee and documented in the committee's 1978 Journal of Development (ref. 5).

The IPIP data manipulation language is based on the data manipulation language specified by the CODASYL FORTRAN Data Base Committee (ref. 6).

The IPIP logical schema and data manipulation languages are compatible, syntactically and semantically, with subsets of their CODASYL counterparts. The IPIP languages contain extensions to enhance functionality or usability; however, where IPIP and CODASYL functionalities coincide, CODASYL syntax should be valid IPIP syntax, and semantics and status return code should be identical.

IPIP compatibility with anticipated standards has several advantages. In the long term, many systems will conform to the standard. IPIP compatibility with standards will facilitate coexistence with, and migration from, these systems. IPIP compatibility with standards will facilitate incorporation of IPIP extensions for engineering and manufacturing into products based on standards and incorporation of these extensions into the actual standards.

Logical Schema Language.- An IPIP logical schema describes logical data structures for a collection of data. A logical schema is written using the logical schema language and is compiled by the logical schema language compiler. The compilation process results in validity checking of the source schema, error reporting, and entry of an encoded representation of the source schema into the data inventory management subsystem (DIMS) data base.

A logical schema models an enterprise in terms of logical data structures. Thus the IPIP logical schema language contains facilities for modeling (1) objects dealt with by the enterprise, (2) attributes of these objects, and (3) relationships between these objects. In this connection, the logical schema language contains facilities for describing comparability of attributes and constraints on and between objects. The logical schema language

provides for the declaration of named keys (composed of one or more attributes) that figure in the declaration of uniqueness constraints and in data manipulation. The logical schema language also provides for the declaration of structures composed of objects and relationships that may be used (along with mapping schema declarations) to describe objects in other logical schemas. In future releases, these structures may be referenced in IPIP data manipulation commands.

Examples of objects modeled by a logical schema dealing with airplane design include (1) airplane performance, (2)airplane configuration, (3)planforms, (4) engine characteristics, and (5) engine performance.

Figure 7 is a data structure diagram that represents these object types by rectangles and relationships between them by arrows. Figure 8 includes occurrence diagrams that describe occurrences of the PERFORMANCE CRITERIA and CONFIGURATION PARAMETERS objects. Rows in a table represent individual objects. Columns in a table represent attributes of these objects. The intersection of a row and a column (a field) contains the value of an attribute for a particular object. An arrow in a data structure diagram represents a relationship between one object of the type at the head of the arrow and one or more objects of the type at the tail of the arrow. The AIRPLANE-CHARACTERISTIC relationship of figure 7 is such a 1:n type of relationship between PERFORMANCE CRITERIA and CONFIGURATION PARAMETERS objects. Figure 8 illustrates this relationship between one STOL type of airplane and one STOL airplane (STOL1) and between one CTOL type of airplane and two CTOL airplanes (CTOL1,CTOL2).

The IPIP logical schema language supports both the network and relational data models and, functionally, the hierarchical data model. The principal difference between the network and relational data models is the style used to describe and manipulate data for the purpose of modeling the enterprise. For example, in the CODASYL network data model, object types are modeled through "record" declarations, and 1:n relationships between "owner" and "member" records are modeled through "set" declarations. One speaks of objects as record occurrences and groups of objects related to each other by a set as set occurrences. Two set occurrences are illustrated in figure 8, one corresponding to airplane type STOL and one corresponding to airplane type CTOL.

In the relational model, there are various terminologies used. For example, object types are modeled through "relation" or "table" declarations. Objects are referred to as "n-tuples" of the relation or rows of a table. IPIP languages use the term "relation" and treat it as a synonym for "record". Traditionally, relationships between object types are uniformly perceived in the relational data model in the implicit fashion of comparing values

of objects with comparable attributes. Thus relationships are not explicitly declared or named.

Airplane-characteristic relationships in figure 8 are determined by corresponding values of attribute AIRPLANE TYPE for PERFORMANCE CRITERIA and CONFIGURATION PARAMETERS in either data model. The difference is that, in the network data model, the user traditionally perceives relationships as something, formally declared and named in a schema, that relates the two attributes; whereas, traditionally in the relational data model, the user perceives only an implicit relationship between objects that is determined by values of comparable attributes. And, traditionally, interrecord data manipulation commands in the network data model reference sets and corresponding commands in the relational data model compare attributes directly.

It has been recognized that there are constraints associated with the network set construct that should be expressible in the relational model. These include 1:n-ness and whether or not an owner must exist when a member is created. It has also been recognized that named structures, such as connected record set paths, facilitate perception of relationships and expession of data manipulation statements. Reference to one named structure in a data manipulation command can be much simpler than stating the conditional expression defining the structure. Thus IPIP takes the relational concept of foreign key that deals with 1:n-ness and provides for declaration of named foreign keys that can be referenced in data manipulation statements. The semantics of the network schema set construct are embodied in the semantics of a schema foreign key construct using syntax that highlights parallel semantics (as a convenience to those who must deal with both data models). The IPIP data manipulation language permits reference to foreign keys.

On the other hand, the ability of relational data manipulation languages to perform interrecord operations in the absence of predeclared relationships is recognized to be very useful. Hence in future releases, the IPIP data manipulation language will permit interrecord operations through comparison of attributes as well as through reference to sets. That is, one might select object occurrences via the AIRPLANE-CHARACTERISTIC set or foreign key or, equivalently, where AIRPLANE TYPE OF PERFORMANCE CRITERIA EQUALS AIRPLANE TYPE OF CONFIGURATION PARAMETERS.

IPIP also provides for the declaration of domains that are offered by some relational systems to establish comparability of attributes. (Some relational systems base attribute comparability on data type comparability.)

The IPIP DATA MODEL schema clause governs whether the schema is network or relational or a mixture of both. If NETWORK is

specifed, then the foreign key construct may not be declared. If RELATIONAL is specified, then the set construct may not be declared. Domain declarations are allowed in either case. Interrecord operations based on attribute comparison are permitted in either case. RECORD and RELATION are synonyms and may be used interchangeably.

Attributes are modeled through data item declarations in the IPIP logical schema language. The data type of a data item may be declared directly in a data item declaration or indirectly through reference in the data item declaration to a domain declaration. Naming conventions and syntax for data item declarations conform to the programming language specified in the schema HOST LANGUAGE clause.

The IPIP logical schema language provides for the declaration of named logical keys composed of one or more data items. Keys may be declared to be unique, although enforcement of this constraint is deferred to a future release. Data manipulation commands may reference keys by name, or may reference data items that are contained in a key declaration. Key items must map to key items in underlying logical or internal schemas. In this way, data manipulation commands are restricted to items that are supported by indexes specified in an internal schema.

The IPIP logical schema language provides for the declaration of structures composed of objects and relationships (records and sets). Mapping schema facilities may be used to map a logical schema record onto a structure-declared record in an underlying logical schema. In future releases, logical schema structure declarations may be referenced by IPIP data manipulation commands.

Logical schema structures, a general-purpose modeling facility, were included in early releases of IPIP to satisfy geometry processing requirements. For example, a SEGMENT logical record in a geometry-oriented schema is mapped to one or more occurrences of related vector and vertex records (and others) in an underlying logical schema. This technique allows a base-level logical schema model of geometry that will eventually support a variety of logical views through interschema mappings. Also, since an occurrence of a user-visible record corresponds to several occurrences of underlying records, a single user-specified data manipulation command against the former is translated into several commands against the latter. More work is performed per user command against the higher-level objects. User productivity and system performance are enhanced accordingly. (See reference 7 for a more detailed discussion of the use of structure for geometry processing.)

Early releases of IPIP software will process structure- and nonstructure-defined records separately. Structure-defined record processing provides some features not provided by nonstructure-

defined record processing. For example, structure-defined record processing supports the CODASYL SOURCE facility for derived items and propagated deletion of records. In keeping with IPIP philosophy, future releases will integrate software and functionality for structure and nonstructure processing.

Future releases of the IPIP logical schema language will provide for declaration and enforcement of constraints on and between objects.

Internal Schema Language.- An IPIP internal schema describes physical data structures to support the logical data structures of a logical schema. An internal schema is written using the internal schema language and is compiled by the internal schema language compiler. The compilation process results in validity checking of the source schema, error reporting, and entry of an encoded representation of the source schema into the DIMS data base.

The internal schema language overlaps that of the logical schema language to the greatest practical extent to minimize the amount of IPIP schema language with which the administrator must deal. Thus the logical and internal schema languages use the same language constructs to declare physical or storage records (relations), data items, domains, and keys.

Constraints and relationships are not declared in an internal schema. Domain declarations are used only to facilitate data item declaration. The comparability semantics of the domain construct do not apply in the internal schema language. Key declarations are limited to one data item in the internal schema language.

The internal schema language also includes facilities for declaring and configuring storage areas and for mapping them onto operating system files. Block sizes for storage areas may be declared along with the initial number of blocks and expansion factors. Also included are facilities for mapping internal schema records (relations) and data item indexes and other supporting physical structures to storage areas.

Mapping Schema Language.- An IPIP mapping schema contains statements that map data structures of a logical schema onto data structures of one or more underlying logical and/or internal schemas. A mapping schema is written using the mapping schema language and compiled by the mapping schema language compiler. The compilation process results in validity checking of the source schema, error reporting, and entry into the DIMS data base of bindings of upper schema constructs to lower schema constructs.

The mapping schema language includes facilities for mapping (1) records/relations to records, relations, or structures; (2) sets/foreign keys to sets or foreign keys; and (3) data items to data items.

The names of records, relations, sets, foreign keys, and data items may differ between upper and lower schemas. Similarly, the host language may differ. The data model may differ between upper and lower logical schemas.

Data Manipulation Language.- IPIP data manipulation commands are expessed in statements of a single data manipulation language that may be used either in application programs or in interactive applications at a terminal.

Data manipulation commands in an application program are processed by a data manipulation precompiler. The precompilation process results in validity checking of data manipulation statements, error reporting, conversion of source data manipulation statements into comments, and generation of host language assignment statements and call statements. The precompiler also generates host language communication area declarations corresponding to record and item declarations in the schemas invoked by the program. The precompiler enters into the DIMS data base a binding of the data manipulation command to the schema against which it is written.

The precompiled program is then compiled using the host language compiler. Data manipulation commands within the program will be fully bound to the internal schema before their execution. Binding options provided by IPIP are discussed in a later section of this paper.

Data manipulation commands in an interactive application are processed by the IPIP query processor. This processing results in validity checking of data manipulation statements, error reporting, full binding of the data manipulation command to an internal schema, and then initiation of command execution.

The IPIP data manipulation language provides the following facilities:

1. INVOKE (or signify) one or more logical schemas against which other commands are written

2. OPEN schemas against which commands are to be executed

3. CLOSE schemas

4. ACCESS (or lock) record types, data sets, and their intersections for retrieval or update operations

5. FIND (or locate)record occurrences using various selection criteria

6. GET (or retrieve) a record located by a previous FIND command

7. FETCH (or locate and retrieve) a record as in a FIND/GET combination

8. STORE a record

9. MODIFY a record or multiple records, as specified by a selection expression

10. DELETE a record or multiple records, as specified by a selection expression

11. REMOVE (or delete) a record or multiple records as specified by a selection expression with restricted propagation of deletion. (REMOVE is restricted to structure-defined records in early releases.)

12. COPY a record producing another which is identical except for unique key values. (COPY is restricted to structure-defined records in early releases.)

CODASYL currency indicators are maintained to support traditional CODASYL style network traversal. Most CODASYL options are supported for the FIND, GET, STORE, DELETE, and MODIFY operations. The FETCH operation is an IPIP extension to CODASYL. The multiple record options for the MODIFY and DELETE operations are IPIP extensions to CODASYL. Multiple record retrieval operations are intended for future releases.

Another IPIP extension to CODASYL is the provision for multiple cursors. Upon execution of a location command (FIND/FETCH) in which a selection expression (WHERE, USING, or KEY phrase) is specified, a find file is constructed and associated with the cursor number specified in the user command. The find file contains locators for qualifying record occurrences. Location operations without selection criteria cause repositioning within the find file associated with the cursor number in the user command. Retrieval operations (GET/FETCH) cause the currently identified record occurrence within the find file associated with the cursor number in the user command to be delivered to the application. The multiple cursor capability permits more than one find file to be maintained concurrently for a single record type. This is very useful in applications such as the parts explosion application, where there is recursive processing on the same record type.

The IPIP data manipulation language supports interrecord operations in the CODASYL network style of referencing sets which

relate records. In future releases, the data manipulation language will support interrecord operations via the relational style of comparing data items. In the examples shown in figures 7 and 8, one might select record occurrences via the AIRPLANE-CHARACTERISTIC set or foreign key or, equivalently, where AIRPLANE TYPE OF PERFORMANCE CRITERIA EQUALS AIRPLANE TYPE OF CONFIGURATION PARAMETERS.

Integration of Functionality and Software

IPIP stresses integration of functionality in terms of (1) uniformity of user languages, (2) commonality of capabilties offered at different user intefaces, and (3) an integrated software architecture. As discussed in previous sections, user languages reflect integration through the unified approach to data models and host languages, a single logical schema language, a single mapping schema language, a single data manipulation language, and commonality between the internal and logical schema languages.

Much of this functional integration is based on the observation of the commonality in functions offered at different levels in multiple-level data architectures. This commonality is obscured in most systems by the use of different languages at each level, e.g., three-schema languages and various data manipulation languages. Also, a particular capability may be found at selected levels in one system and at different levels in another system.

The example of figure 9 illustrates functional commonality across the IPIP data architecture. This commonality is exhibited by the storage schema/schema/subschema/ DML data architecture of CODASYL. At each level of rectangle, including that of data manipulation (application), a record is declared. And at each level of diamond, including that of data manipulation, the same type of mapping (projection) is used.

Record EP2 at the base logical level is the ENGINE PERFORMANCE record of figure 8. It is the projection of data items 3, 4, 5, 6, 7, and 8 of record EP1 at the internal schema level. Note that EP1 contains two implementation-oriented data items (RECORD CODE and RECORD LENGTH). Similarly, record EP3 is the projection of data items 1, 2, 3, and 4 of EP2. And, at the application level, a data manipulation command requests data items 1, 2, and 4 of EP3. This is equivalent to implicit declaration of a record, EP4, which is the projection of data items 1, 2, and 4 of EP3.

It is usually the case, as in this example, that a function available at one level of the data architecture is useful at the others. IPIP takes symmetry of functional capabilities across the data architecture, to the extent useful, as a principal objective.

Functional symmetry promotes convenience, because the user does not remember which functions are available at which interfaces. Functional symmetry offers other advantages. For example, it would have been possible to obtain EP4 through a projection of data items of EP2. However, through the use of projection between logical schemas, the person developing the application is shielded from unnecessary details (a simplification) and potential security problems are avoided.

IPIP takes advantage of functional symmetry through use of a highly integrated metadata data base (the DIMS data base) and software architecture. The software architecture is described in a later section of this paper. Figure 10 presents a simplified abstraction of the DIMS data base to illustrate IPIP strategies for functional and software integration.

In figure 10 we see in the top row of circles that the declarations of records EP1, EP2, EP3, and EP4 are represented in DIMS in exactly the same way. The dotted circle for EP4 indicates that in the first release IPIP constructs the metadata describing the declaration of EP4 but does not store it on DIMS. Metadata in the first row is generated through compilation of an internal or logical schema or by the processing of a data manipulation command by the precompiler or query processor. Since the metadata is identical in each case, common semantic routines are used by the internal and logical schema compilers, the precompiler, and the query processor to support declaration.

The second row of circles illustrates metadata that links or binds adjacent levels in the data architecture. (The example of figure 9 is used here.) The dotted circle in the second row indicates that this binding information is never actually constructed. Circles in the second row are generated by compilation of a mapping schema or by the processing of a data manipulation command by the precompiler or query processor. Since this binding metadata is the same in each case, common semantic routines are used by the mapping schema compiler, the precompiler, and the query processor to support mapping.

The circles in the third row represent binding information identical to that in the second row. Circles at this level represent the composition of two adjacent mappings and therefore bind alternating levels in the architecture. Binding software merges two second-level circles to obtain a third-level circle. All third-level circles are built by the same binding routines. Similarly, the same binding software is used to produce the binding information in the fourth row.

The use of common semantic routines and common binding routines promotes compactness in the IPIP software supporting an ANSI-like three-schema data architecture, while providing support

(through repeated application) for the generalized n-level IPIP data architecture.

The uniformity of DIMS tables permits the use of common software for other functions. For example, software that reports where a declaration at level m in the architecture is used at level m+1 may be used for any two adjacent levels. This is true even for the application level if application metadata is stored in DIMS.

Reference 8 discusses integration of functionality and its application to design of software for data base management systems.

Binding

In a multiple-level data architecture, a data manipulation command must ultimately be resolved in terms of underlying schema declarations before it can be executed. One may think of the process of binding as one of recursive substitution of declarations. Binding of a data manipulation command against a logical schema record to an internal schema permits translation at execution time of this command into one or more operations against internal schema records.

As discussed in the previous section, the circles in rows 2, 3, and 4 (fig. 10) represent DIMS metadata binding together different levels of the data architecture. The second and third rows represent bindings of adjacent and alternating levels, respectively, in the architecture and are called partial bindings. The circle in the fourth row represents metadata that fully binds a data manipulation command to an internal schema. The previous section describes the use of common software for generating binding metadata.

Binding in a multiple-level data architecture is expensive. Therefore, IPIP supports several binding options. IPIP permits partial or full binding and run-time or prerun-time binding and combinations thereof. EP3 could be bound to EP1 prior to developing the application that includes EP4. If EP4 is issued interactively, the rightmost binding circle at the second level would be generated and merged at run time with the leftmost circle at the third level to fully bind EP4 to EP1. If EP4 is in an application program in test mode, the same strategy might be used. If the application program is in production mode, the data administrator would probably fully bind EP4 to EP1 before program execution.

Note that given prerun-time binding of the uppermost logical schema to the internal schema, full run-time binding of the data

manipulation command is a two-level operation, regardless of how many levels of underlying logical schemas there are.

IPIP SOFTWARE ARCHITECTURE

IPIP prototype software is a collection of software subcomponents designed as a nucleus around which the full IPAD data management functions can be built. This section of the paper discusses the IPIP software architecture and traces a user data manipulation command through the system.

There are three major sets of software subcomponents in the architecture: IPIP user interfaces, stubs, and IPIP data manager. IPIP user interface subcomponents are used by end users when defining data structures and when formulating commands against the data base. These subcomponents include the query processor, precompiler, logical schema compiler, internal schema compiler, and mapping schema compiler. IPIP data manager subcomponent routines execute end user requests against the data base and provide services supporting IPIP user interface subcomponents. The stubs subcomponent is linked to user interface subcomponents and to all user data management application programs. Figure 11 shows diagrammatically how the various subcomponents fit in the data management architecture.

Some of the subcomponents in the diagram (i.e., the prototype implementation of the IPAD executive (IPEX) routines) are not discussed in this paper. For purposes of this paper, these subroutines are broken into two groups, the first group providing network services and referred to as the IPSR routines, the second referred to as the IFCS routines and providing a host-independent file system to IPIP. A complete discussion of their design can be found in reference 9.

Before the IPIP software modules are described in more detail, several features of the software architecture should be mentioned. The features listed below have had a direct influence on software and on the software architecture. Features fitting into this category include multi-user/multithread processing, data set processing, versioned data processing, recursive data dictionary (DIMS), and adaptability of code to different types of machines.

The following is a brief description of the various software subcomponents in the IPIP software architecture.

IPIP User Interface Subcomponents

Precompilers.- The precompilers convert programs containing embedded data manipulation commands into programs containing host-language-compatible requests to the data manager. The conversion process utilizes the data inventory management subsystem (DIMS) data base in the editing, validation, and generation of requests to the data manager. The prototype precompilers generate code for FORTRAN and Pascal programs as well as declarations of communication areas for staging records invoked by the program.

Query Processor.- Like the precompilers, the query processor accepts data manipulation requests and produces requests for data from the data manager. However, the query processor is used only for interactive requests, thereby eliminating the need for source code generation. The query processor performs syntactic checks on the request and transmits it to the IPIP data manager, where the command is checked semantically and then bound and executed. Upon completion of the processing in the IPIP data manager, the query processor displays the results on the user's terminal, a local file, a line printer, or a combination of the three. With a couple of minor exceptions, the data manipulation language processed by the query processor is identical to that used by application programs.

Compilers.- The logical, internal, and mapping schema compilers all perform the same functions but for different languages. The compilers perform syntactic checking of source schema declarations and send encoded versions of them to the IPIP data manager. When the IPIP data manager completes the processing of a declaration, it returns status to the compiler, which passes it back to the user.

IPIP Data Manager Subcomponents

The IPIP data manager subcomponents comprise the portion of the data base management system that carries out data manipulation, binding, and semantic processing actions requested by users. The IPIP data manager contains ten subcomponents: the scheduler, message procedure interface processor (MPIP), common semantic processor (CSP), data base control subsystem (DBCS), data manipulation subsystem (DMS), record translator, binder, presentation services, access module (AM), and resource manager (RM). Each of these subcomponents is discussed briefly.

Scheduler.- As the name implies, this component schedules requests for execution or restarts requests that were suspended before completion. Other functions performed by this component include,

but are not limited to, initialization of IPIP, defining Pascal work areas for each thread, and ensuring that the proper code is in memory when switching between threads.

Message Procedure Interface Processor (MPIP).- The major function of this subcomponent is to unpack incoming user requests and to package outgoing messages for shipment across the network. Incoming messages are routed to either the DBCS or the CSP routines, depending on function codes sent with the message. MPIP requests a binding record identified by the encoded version of a user command.

Common Semantic Processor (CSP).- The common semantic processor performs all semantic actions requested by the precompiler, query processor, and the schema compilers. The CSP is invoked by MPIP and recursively calls IPIP for its data management needs. Common routines within the CSP perform equivalent semantic processing across compilers.

Data Base Control Subsystem (DBCS).- The major function of the DBCS is to process user-level data manipulation requests. The DBCS receives an encoded version of the user request in the form of a binding record and converts this request into primitive operations provided by the binder, DMS, record translator, resource manager, and presentation services. The DBCS supports the retrieval, storage, and update of data through the use of a single data manipulation language, which provides network facilties (CODASYL based) and relational facilities. The data manipulation language also supports the manipulation of geometric entities such as lines, points, composite curves, and surfaces. Inputs to the DBCS consist of a binding record, which is the encoded data manipulation command including attendant meta information, and a job status record, which contains the current execution status of the user making the request.

Data Manipulation Subsystem (DMS).- The DMS contains a set of primitives that operate on internal-schema-defined data records. It is these primitives to which each user-level data manipulation command is decomposed. DMS is another subcomponent that makes user-level data manipulation requests to IPIP for some of its run-time data.

Record Translator.- This subcomponent converts records from logical views to internal representations and vice versa. These conversions include changes in both the position of fields within the records and changes of data types and sizes of fields.

Binder.- This subcomponent reconciles differences between multiple views of the data. In other words, it creates metadata (binding records) specifying how to translate from a user operation on a logical data structure to one or more internal requests on the physical data structures described by an internal schema. The binder can be invoked either prerun-time or when the command is executed (run-time). The binder may bind either a logical schema or a program or query to an underlying internal schema.

Presentation Services.- In prototype IPIP, the only function of this routine is to log IPIP errors on a system log. As development continues, the set of routines will be expanded to include screen formatting logic for various types of terminals.

Access Module.- This subcomponent creates and maintains B-trees used to index physical data files. The access module performs B-tree look-up to satisfy all user requests for data.

Resource Manager.- This subcomponent provides service routines used by all IPIP data manager subcomponents for resource-related services. Routines are provided to perform buffer management, record management, and file management services.

Stubs.- This subcomponent provides service routines that link users with IPIP. All user programs and the IPIP user interface subcomponents have a copy of these routines linked to them. Stubs packages and unpackages information to be transmitted or received via communications software and calls IPSRs to initiate communication.

Trace of User Query

The following is a high-level trace of a user query through the data base management system. A STORE command was chosen because it shows the processing flow of a representative user-level request.

The following assumptions should be noted before reading the example:

1. Internal data manipulation requests will be identified but not traced.

2. Logical and internal schemas have been defined and are mapped to each other.

3. All IPIP initialization has been completed.

4. The user has opened and locked the record type to be stored.

5. This command is the only one executing in IPIP.

6. MPIP is ready to receive the message.

The process starts when the query processor recieves a command from a user. The query processer performs syntactic checking on the command and breaks out individual item or record names within it. These names are sent through stubs, the IPAD data manager, and network services to the CSP routines in the IPIP data manager. (The mechanisms of this network transfer are discussed below.) The CSP performs semantic checking to validate the command. During this process a binding record is built linking the DML command to the correct logical schema. After the validation has been completed and the binding record built, status information is sent back to the query processor through the network. The query processor receives this status information and, assuming the request was valid, builds a collection of tables that contain information from the user request. Stubs is then called to transmit these tables to IPIP and, using IPSR data translation routines, requests that these tables be converted from a machine representation to the network standard representation. After the translation has been completed, the network IPSR routines are called to send the tables across the network to the IPIP data manager.

MPIP reads this message from the IPSR routines and decodes the control information associated with the command. Again using the IPSR routines MPIP then translates the tables into IPIP standard form. After the data has been translated, the binding record created during semantic validation by the CSP is augmented with run-time information supplied in the command.

The binding record is now in a form that can be processed by the DBCS and lower-level routines. However, before the DBCS is called, the job status record (JSR) is retrieved. This record contains the current status of the user and is used extensively by the DBCS. This JSR record is retrieved by MPIP using a user-level data manipulation request.

Upon receipt of the binding record and job status record, the DBCS begins processing the request, determines if binding has been completed, and, if not, calls the binder to complete the binding process.

The binder accesses binding records associated with this request and creates a binding record that fully binds the user command to an internal schema. Having completed its processing, the binder returns this binding record to the DBCS, which can now continue processing. Space is allocated, using resource manager routines, for the internal record (or records) that is to be added

to the data base. The record translator is called to create an internal record (or records) from the user-supplied external record in the space allocated above. After creation of the internal record, the data manipulation subsystem is called to add the record (or records) to the data base. Before the record (or records) is added, the open status of the data files is verified. This verification is performed by making user-level DML requests to the DIMS tables containing authorization information. If the user passes these checks, the DMS calls the record manager routines in resource manager to add the data to the data file and to add other information to other data base files. The resource manager calls the IFCS to perform the physical write. DMS then calls the access module to update indexes on the keyed fields in the record. These indexes are represented using B-tree data structures.

The data has now been added to the data base; all that remains is to return through the routines and return status informing the user of the successful completion of his command.

REFERENCES

1. Southall, J. W.: NASA Contractor Report 2984, Integrated Information Processing Requirements, March 1979.

2. Comfort, D. L.: NASA Project Explores DBMS in Aerospace Use, Computer World, p. 46, November 12, 1979.

3. Tsichritzis, D. and Klug, A. ed.: The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems, AFIPS Press, 1977.

4. Larson, J. A., Johnson, H. R., and Wilson, T. B.: A Common Database Architecture for Distributed Processing, Hawaii International Conference on System Sciences, Honolulu, Hawaii, January 3-4, 1980.

5. CODASYL Data Description Language Committee Journal of Development, January 1978.

6. CODASYL FORTRAN Data Base Committee Journal of Development, August 1979.

7. Dube, R. Peter, Herron, Gary J., Schweitzer, Jean E., and Warkentine, Erich R.: An Approach for Management of Geometry Data, IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 12 of this compilation.)

8. Johnson, H. R., Larson, J. A.: Data Management for Microcomputers, IEEE Compcon Fall 79, p. 191-194, Washington, D.C., September 4-7, 1979.

9. Tanner, J. G., Kirkwood, D. M., and Ives, F. M.: Executive and Communications Services To Support the IPAD Environment, IPAD: Integrated Programs for Aerospace-Vehicle Design, NASA CP-2143, 1980. (Paper 10 of this compilation.)

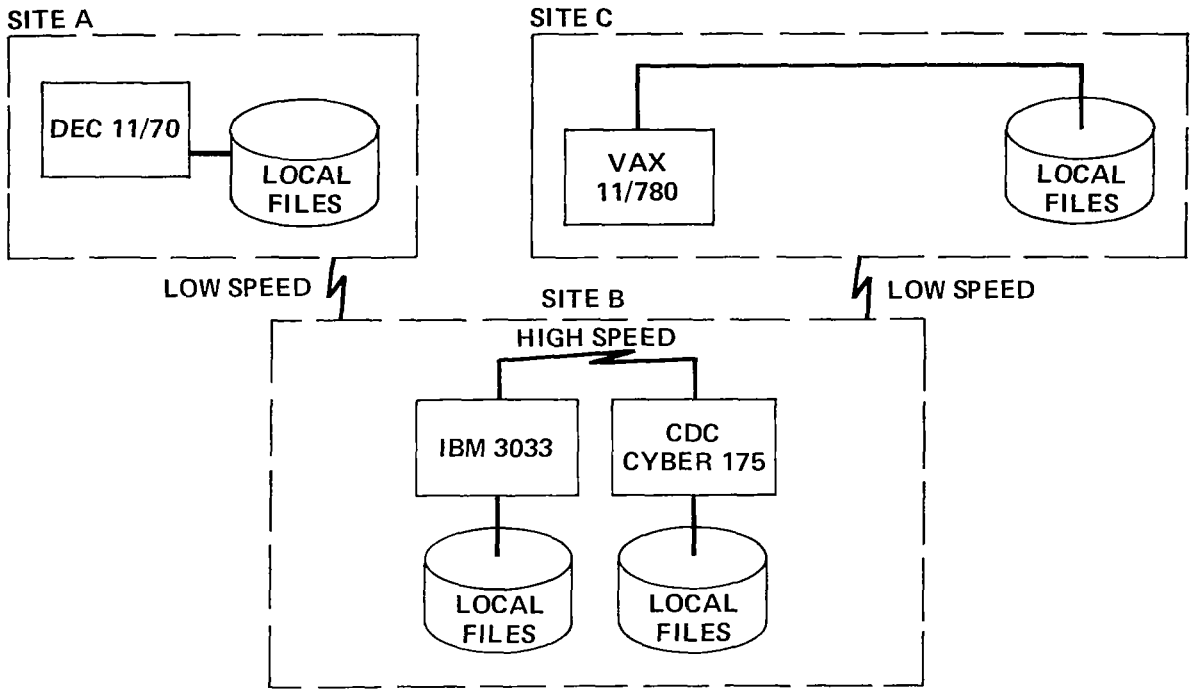


Figure 1.- Sample configuration for distributed processing.

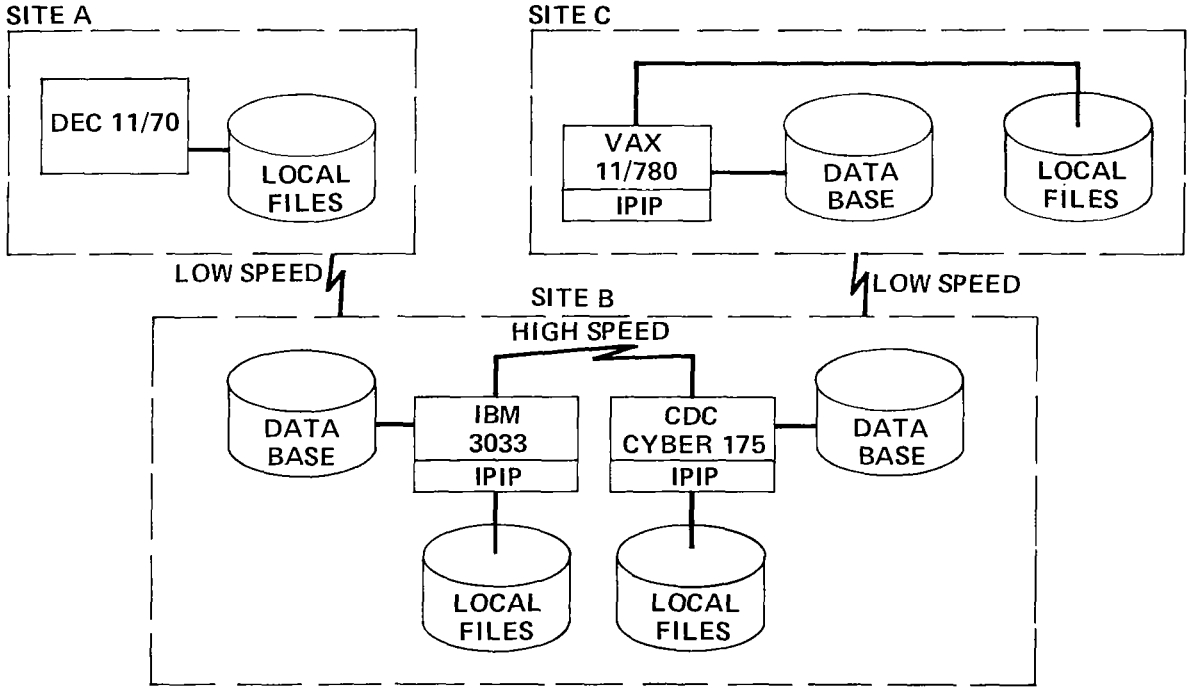


Figure 2.- Multiple IPIPs in a distributed configuration.

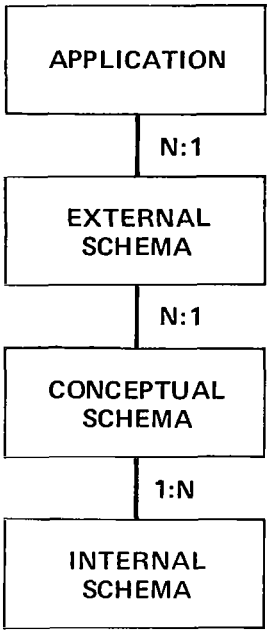


Figure 3.- ANSI/X3/SPARC DBSG data architecture.

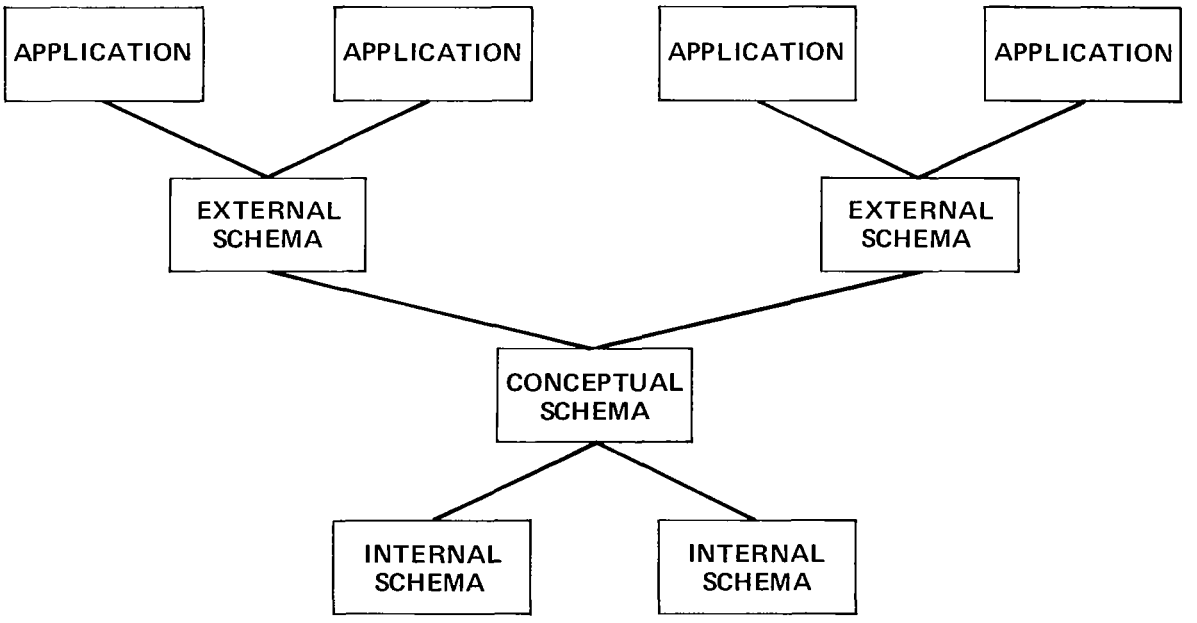


Figure 4.- Typical ANSI arrangement of applications and schemas.

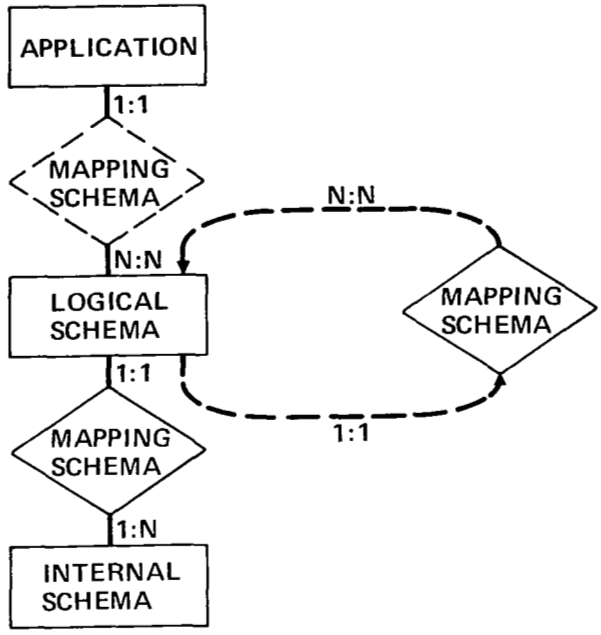


Figure 5.- IPIP data architecture.

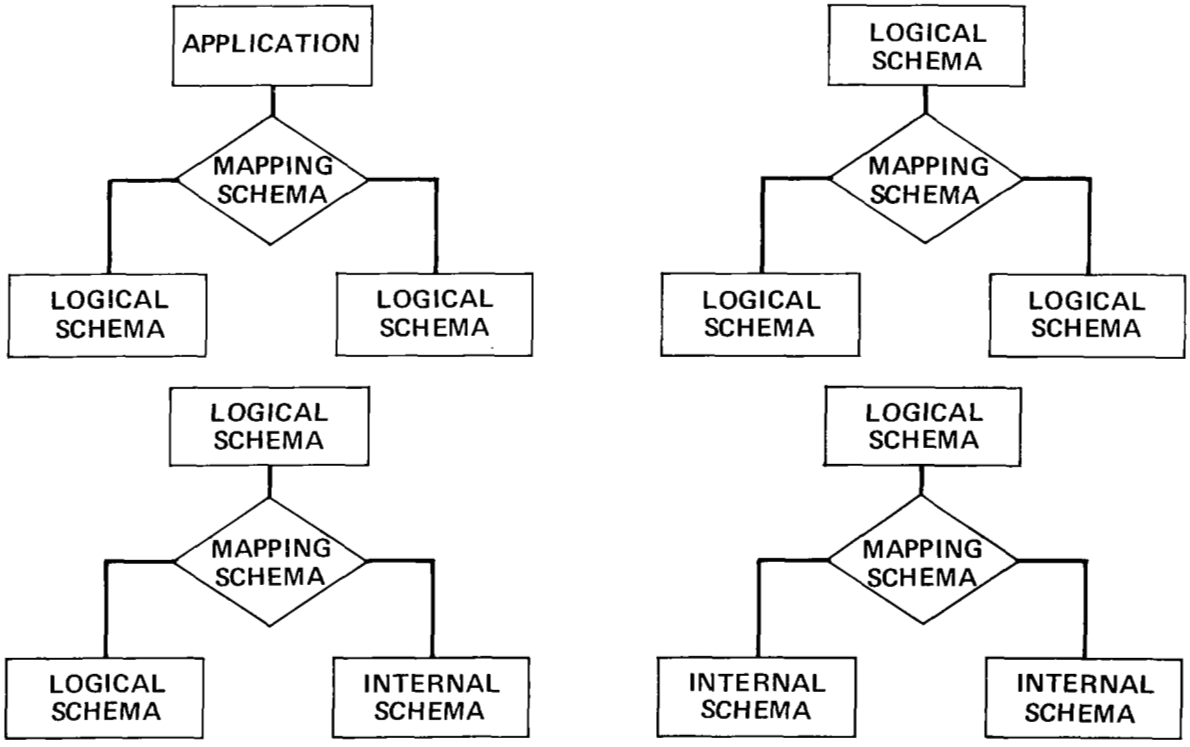


Figure 6.- IPIP arrangements of applications and schemas.

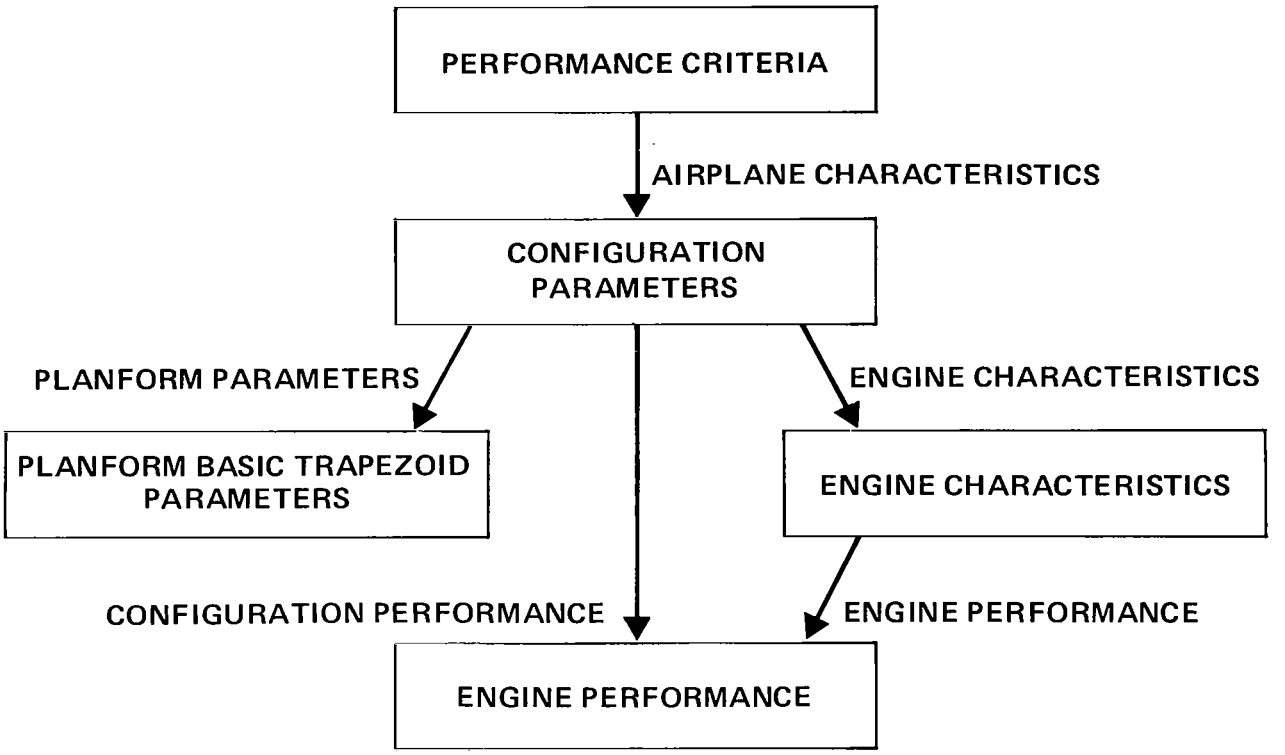


Figure 7.- Logical schema data structure diagram.

PERFORMANCE CRITERIA (PC)

AIRPLANE TYPE (IAT1)	PAYLOAD TYPE (IPT1)	RANGE CLASS (IRC1)	DESIGN PAYLOAD PASSENGERS (DPP)	DESIGN PAYLOAD CARGO (DPC)	GROWTH PAYLOAD PASSENGERS (GPP)	GROWTH PAYLOAD CARGO (GPC)	PASSENGER WEIGHT ALLOWANCE (PWA)
STOL	PASSENGER/ CARGO	SHORT	135	5000	170	10000	165
CTOL	PASSENGER	MEDIUM	175	5000	10000	225	205
CTOL	PASSENGER	LONG	200	5000	250	10000	225

CONFIGURATION PARAMETERS (CP)

MODEL NUMBER (MN5)	AIRPLANE TYPE (IAT5)	PAYLOAD TYPE (IPT5)	RANGE CLASS (IRC5)	NUMBER OF ENGINES (NE)	ENGINE MODEL NUMBER (IEMN5)	ENGINE LOCATION (IEL)	MARKET CATEGORY (IMC)	CUSTOMER (IC)
STOL1	STOL	PASSENGER/ CARGO	SHORT	2	CF6-50A	WING	TRANSPORT	USAF
CTOL1	CTOL	PASSENGER	MEDIUM	2	CF6-6D	BODY	DOMESTIC	UAL
CTOL2	CTOL	PASSENGER	LONG	4	JT9D-3	WING	INTER-NATIONAL	JAL

ENGINE PERFORMANCE (EP)

ENGINE MODEL NUMBER (IEMN4)	RATING ALTITUDE (RA)	CONTINUOUS THRUST (CT)	CONTINUOUS SPECIFIC FUEL CONSUMPTION (CSFC)	CRUISE THRUST (CRT)	CRUISE SPECIFIC FUEL CONSUMPTION (CPFC)
CF6-6D	35000	9700	.639	9060	.631
CF6-50A	35000	11500	.664	10800	.654
CF6-50A	0	48400	.389	42200	.386
CF6-6D	0	39300	.354	37500	.351

Figure 8.- Record occurrences.

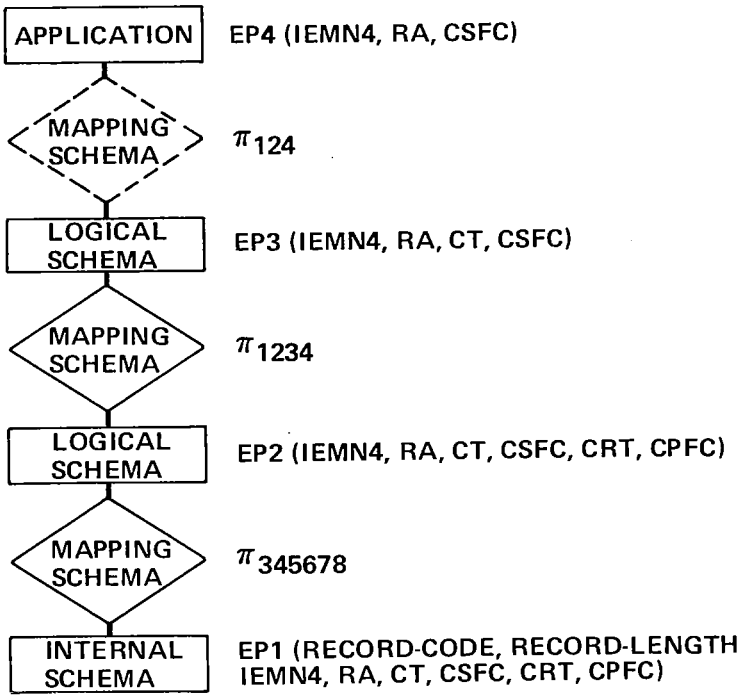


Figure 9.- Symmetry across IPIP data architecture (projection).

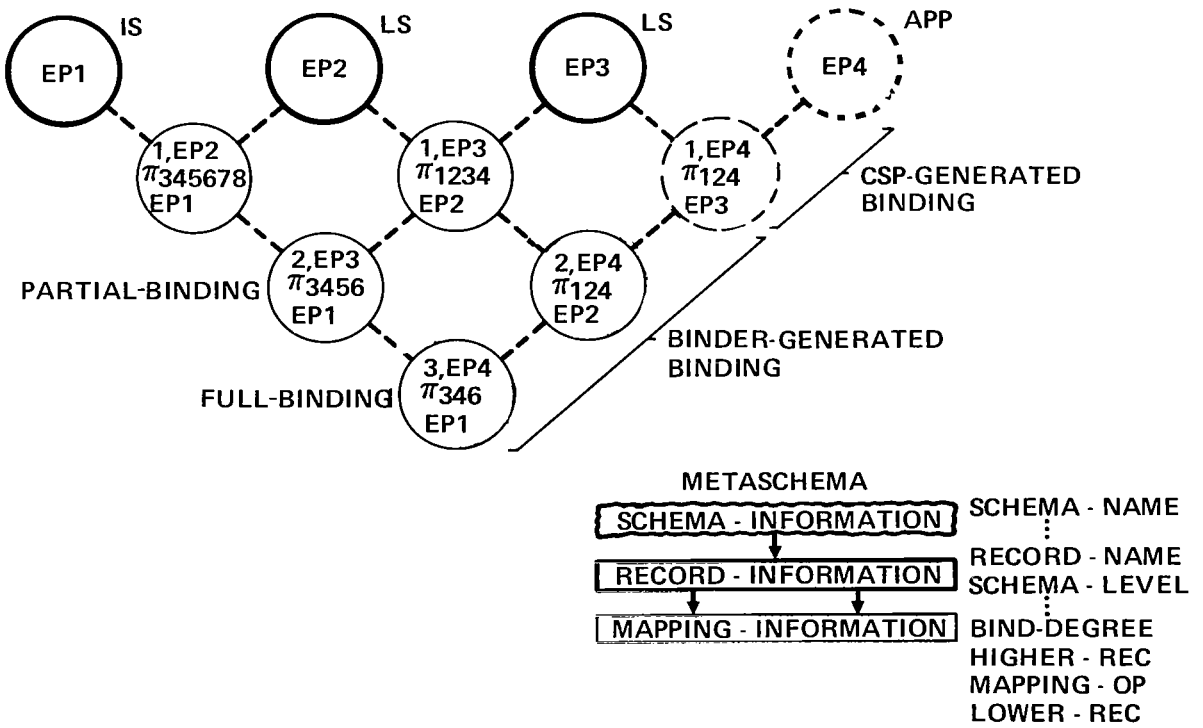


Figure 10.- Symmetry of IPIP metadata and software.

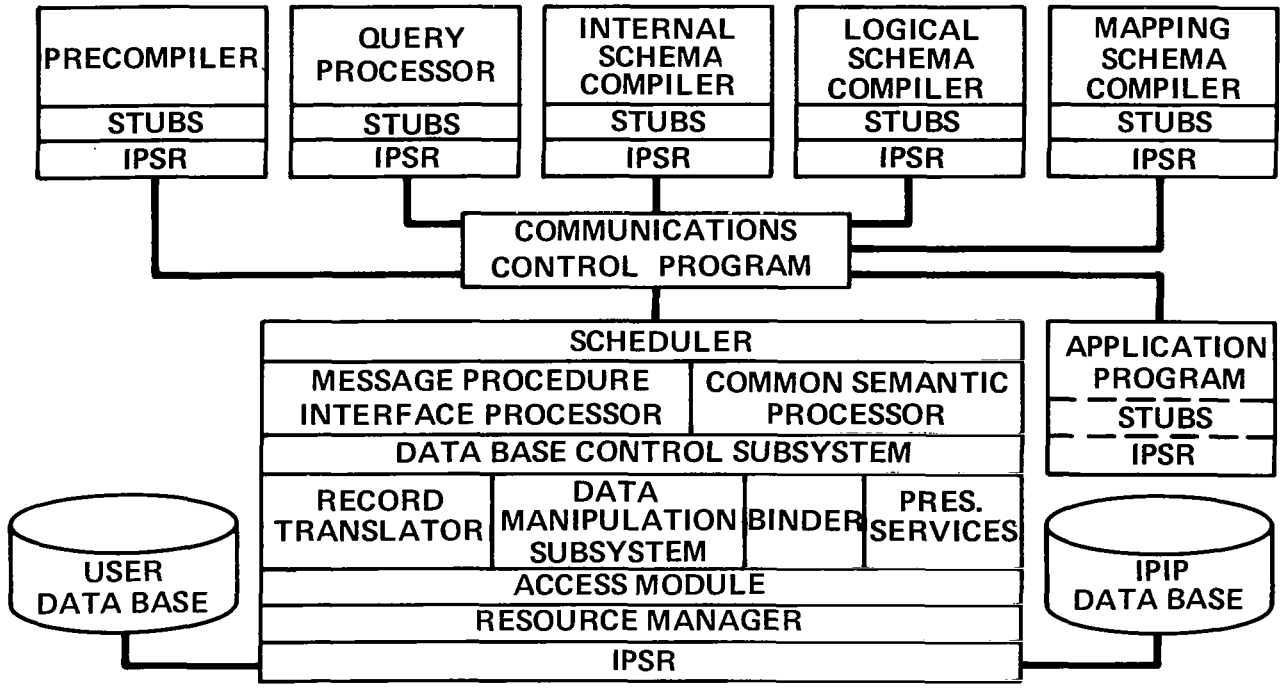


Figure 11.- IPIP components with CCP, IPSRs, and data bases.

AN APPROACH FOR MANAGEMENT OF GEOMETRY DATA

R. Peter Dube, Gary J. Herron,
Jean E. Schweitzer, Erich R. Warkentine
Boeing Computer Services Company

SUMMARY

This report describes the strategies for managing IPAD computer-based geometry. The computer model of geometry is the basis for communication, manipulation, and analysis of shape information. IPAD's data base system makes this information available to all authorized departments in a company.

A discussion of the data structures and algorithms required to support geometry in IPIP (IPAD's data base management system) is presented. Through the use of IPIP's data definition language, the structure of the geometry components is defined. The data manipulation language is the vehicle by which a user defines an instance of the geometry. The manipulation language also allows a user to edit, query, and manage the geometry.

The selection of canonical forms is a very important part of the IPAD geometry. IPAD has a canonical form for each entity and provides transformations to alternate forms; in particular, IPAD will provide a transformation to the ANSI standard. The DBMS schemas required to support IPAD geometry are explained.

INTRODUCTION

One of the objectives of IPAD (Integrated Programs for Aerospace Design) is to develop a computer-based engineering complex which automates the storage, management, protection, and retrieval of engineering data. This complex serves as the central communication integrator for a large number of designers conducting a broad range of design tasks. The cornerstone of the IPAD concept is its information processor (IPIP). IPIP is a data base management system (DBMS) that manages the data required by and produced during the design process.

The design process, as well as interaction with maufacturing and other organizations, has been defined (refs. 1 and 2). It is within this design process that existing computer-aided design (CAD) systems create a machine-readable data base of design information. This data base normally includes not only the specification of geometry but also such information as object mass properties and dimensions. From the sparse definition of objects

for aerodynamics and finite-element analysis to the detail design of parts, geometry is the continuing theme.

Unlike other systems in which geometry information is contained in files accessible only through a design drafting system (or other specialized computing system), the main repository of all information (including geometry) within IPAD is IPIP, the DBMS. With this approach, all tasks, including analysis and design, have access to all of the required data. While a specific CAD/CAM system would nominally be used to create design information, other systems will have controlled access to that data as well (fig. 1).

The principal advantage of the IPAD approach is that the data base management system can support a unified description, manipulation, and management of the data of the organization. The result is a reduction in the duplication of design data, which minimizes problems in maintenance of data base consistency and update efficiency. The shared data base provides a common interface, which aids the integration of various design activities and computer-based support systems. While performing its information storage and retrieval functions, the system can also maintain data base integrity and enforce organization security rules.

The processing and management of geometry in IPAD has resulted in some unique facilities in IPIP. The system must manage geometry, enforce constraints, maintain continuity, and track associations at the primitive level. It must also allow manipulation at the primitive, composite, and object levels of the geometry hierarchy (fig. 2). The system described in this paper satisfies these requirements.

The strategies employed make available to the user data structures that support the management of geometry. In many cases the data structures are independent of the type of geometry being used and, hence, may be employed with different mathematical representations. When IPIP facilities are used with geometric representations (other than IPAD geometry), the manipulation capabilities will be restricted. With the appropriate schema, however, geometry described with a different mathematical form may still be managed at the entity level. The remaining sections of the paper will develop IPAD geometry, geometry data management, and supporting schema development.

IPAD GEOMETRY

Geometry information is stored as collections of components in the data base. An individual component is called an entity. The description of a geometry entity in IPIP is composed of four parts:

Name of entity

Defining values

Mathematical process that produces a shape

Association between entities

The defining values depend on the mathematical form. IPIP uses values directly related to the shape of an object, i.e., positions, tangents, and higher-order derivatives. When that type of data is not available for a specific mathematical form, related coefficients are used.

The mathematical process is a rule explaining how the values are to be used. This rule indicates how to interpret the data; for most representations, however, it is a mathematical function that produces a shape. The definition of this process is not contained in IPIP but is implied by the IPAD canonical form.

The last part of the representation serves to identify the association between entities. In most cases, the dependencies are tracked implicitly by the data structure constructs in IPIP.

The base level building block for the description of geometry in the data base is the vector, e.g., x, y, z. The vectors are found on the terminal nodes of the geometry hierarchy (fig. 2). Entities in the hierarchy are classified as primitive or composite. The primitive entities contain geometry presently in use: points, lines, cubic and conic segments, surface patches, surface of revolution, lofted surface, etc. These pieces of geometry, including composite entities, may then be grouped into objects.

IPAD geometry uses bounded geometry in parametric form. The general form of a geometric entity is given by

$$X = (F_1(t_1,\ldots,t_k), \ \ldots \ , F_n(t_1,\ldots,t_k)).$$

For n=3 and k=1, X is a space curve:

$$X = (F_1(t_1),\ F_2(t_1),\ F_3(t_1)) \qquad t_1 \varepsilon [a,b]$$

when X is a surface, n=3 and k=2:

$$X = (F_1(t_1,t_2), F_2(t_1,t_2),\ F_3(t_1,t_2))$$

$$t_1 \varepsilon [a,b] \text{ and } t_2 \varepsilon [c,d].$$

Each function F_i is defined by user specified values. The meaning of these values in the context of F_i is determined by the IPAD standard form, which was chosen to facilitate management of the geometry entity. In general, the values required for each entity are directly related to the geometry of the entity, e.g., position, tangent, etc.

IPAD Canonical Form

A canonical form, in the IPAD context, is the standard form that the IPAD data management system understands and uses to manage geometry information. The IPAD canonical form is based on parametric representation with a two-point Taylor expansion (also referred to as parametric Hermite interpolation). We now consider the general theory of Hermite interpolation and parametric representation.

Parametric curves and surfaces lead to powerful models that may be used in a computer-aided design system. By parametric representation we mean that each coordinate of a position on a piece of geometry is represented as a function of one or more independent parameters. For example, a curve of one parameter has its position vector on the curve fixed by one parameter. If the parameter is taken from the interval (a, b) and the curve is in space, then the locus of points on the curve, C1, is given by

$$C1 = \left\{ (x,y,z): x=f_1(t), y=f_2(t), z=f_3(t):\ t \varepsilon [a,b] \right\}$$

General theory of Hermite interpolation involves mathematical functions which capture positions and higher-order derivatives at the ends or corners of the pieces of geometry. These formulae are found in Davis (ref. 3).

The canonical form for each IPAD entity can be found in the IPAD documentation. We now give an example of one entity, the cubic segment, using the Hermite strategy. The defining values of the cubic curve $P(u) = (x(u), y(u), z(u))$, are $P(0)$, $P(1)$ (values at the end points) and $DP(0)$, $DP(1)$ (end tangent vectors). The canonical form is given by

$$P(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} D \begin{bmatrix} P(0) \\ P(1) \\ DP(0) \\ DP(1) \end{bmatrix}$$

where

$$D = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

If the matrix multiplication is performed from left to right, the result is the Hermite representation:

$$\begin{bmatrix} A0(u) & A1(u) & B0(u) & B1(u) \end{bmatrix} \begin{bmatrix} P(0) \\ P(1) \\ DP(0) \\ DP(1) \end{bmatrix}$$

Only the values $P(0)$, $P(1)$, $DP(0)$, $DP(1)$, along with the parametric space information, are stored in the data base. Most of the basic enities for both curves and surfaces are handled in a similar manner.

ANSI Communication Standard

The proposed ANSI Y14.26.1 standard, "Digital Representation of Physical Object Shapes," addresses the problem of machine-to-machine dialogue. Its purpose is to facilitate the communication of object shape description messages between data bases of the same company or between intercommunicating companies. The ANSI standard assumes the existence of a geometric shape and proposes standard formats for communicating that shape.

The work of the ANSI committee is and will continue to be an important factor in the definition of IPAD geometry. In some

cases, the IPAD representation of geometry will differ from that of ANSI. This difference is due largely to the fact that IPAD is involved with the management of geometry rather than communication. The IPAD geometry representation contains continuity constraints, association between elements, and other design-related information that will be included in the IPAD format.

IPAD supports the geometry entities found in the ANSI standard. The choice of the defining values for some geometry entities differs from those proposed by ANSI. The system, however, provides transformations to interchange its defining values with those of the ANSI standard.

GEOMETRY DATA MANAGEMENT

This section outlines the approach used in the application of data base management to geometry. This discussion emphasizes features of the IPIP data manager that were developed to support geometry data management requirements. A more general discussion of IPIP is found in the proceedings.

The approach used in the present system differs from that which has been utilized to date. For example, one mechanism for providing geometry data management is to build a set of procedures on top of an existing data manager. This approach was used successfully to construct the IPAD prototype geometry data managers. However, this removes integrity enforcement from the data manager, requires a fixed data format for geometry data, and allows no mechanism to extend the types of geometry that may be stored in the data base.

The approach used in the present system allows the user to define a geometry model by writing a schema. The operations and constraints on the geometry are driven by the schema definitions themselves; thus constraint checking is done within the data manager. The records used by application programs or CAD/CAM systems are defined by logical schema definitions; thus their format may be changed by the data base administrator (DBA). The schema is extendable, so new types of geometry may be added to the system by new schema entries.

The types of schemas necessary to define geometry are determined in part by the data architecture of the IPIP system. First, an internal schema is defined, specifying the organization of data as stored on physical I/O. Second, a base-level schema is required, describing the data of the entire organization (e.g., the IPAD geometry forms used by all the applications of the company). Third, upper-level schemas are defined, providing views of the data necessary for access from application programs or

query users. Data in each schema is related to other schemas using mapping statements in the mapping schema definition language. The data base administrator (DBA) may define an upper-level schema on top of another upper-level schema, thus making n levels of schema definition possible.

The approach used for IPAD/IPIP geometry data management may be summarized as follows:

1. The defining data for the IPAD canonical forms is described in a base-level logical schema.

2. Associations between data items of different records are defined explicitly in the schema.

3. The records of the base-level logical schema are grouped by an explicit schema construct called a "structure".

4. Access and manipulation of a geometric entity are made by DML operations on an upper-level schema record that is mapped from a base-level schema structure.

5. The mapping schema provides the correspondence between items in an upper-level schema and items within record occurrences of the structure.

6. Constraints on the data items and their associations are defined in the schema and enforced during data manipulation.

7. DML on upper schema records mapped from structures has semantics that are consistent with requirements for geometry data management and reflect the underlying structures.

8. DML processing directives are used to direct command-specific semantics.

9. Additional user features are included within the system to aid the effective use of the system and decrease the probability of errors.

Each of these nine concepts is expanded in corresponding sections below.

1. Base-Level Schema Records

The records of the base-level schema contain all the data necessary for the information processing needs of the organization. Nominally, one internal record type exists for each base-level schema record. This internal record has a format describing the way in which the base-level schema record is stored.

While application programs require the geometric entity to be viewed as a unit, with all defining data grouped into a single record, this type of grouping is not suitable for the base-level schema. In the base-level schema, geometric entities are composed of records divided (normalized) to provide the minimum amount of data redundancy. This normalization is achieved in our geometry schemas by placing indirect references to data in place of actual values when those values appear in more than one record type. (See reference 4 for a more detailed description of the normalization process.)

2. Data Associations

Given a set of base-level schema records containing geometry, the IPIP system provides a set of tools that allows the user to describe the relationship of data items occurring in different records. The most important concept used to define these relationships is the CODASYL set construct, which allows the data base designer to specify 1-to-n associations between record types. (A set occurrence is an association of one owner record occurrence with n member record occurrences.) This association is made by specifying matching set selection items in each record.

The integrated data modeling approach within the IPIP system allows the designer to specify these associations using the network or relational style of data modeling. In the network style, associations are described using the CODASYL set construct. In the relational style, the designer can equivalently use the foreign key declaration.

3. Structure Definitions

The structure definitions of the IPIP data base system provide the mechanism for grouping a collection of record types (and occurrences) that are to be accessed and modified as a unit. The structure defines an ordered path through the records that is used by the data base system to process each data base request. This path is based on the associations (sets) that have been defined in the schemas. The first record of the path is called a root record, and it determines a structure occurrence.

The inclusion of the structure facility permits the query or batch user, with one command, to access data within records linked together in complex data structures. The structure facility provides the system with a mechanism to allow an upper-schema record to be composed of items from several lower-schema records.

4. Upper-Schema Records

The upper schema is a description of the data as seen by the application (or CAD/CAM) systems wishing to access the data. These records are stored so that a single DML command may store, modify, or retrieve a geometric entity as a unit. Thus, the user (a CAD/CAM application program or a query user) may access a geometry entity without performing a series of DML commands.

5. Mapping Base-Level to Upper-Level Schemas

The schema-mapping facilities within IPIP that are used to map items of the upper-schema records from items of records in a structure provide a facility not found in most data base management systems.

The DBA may specify a mapping not only from an item within a record type, but also a mapping from an item of a particular occurrence of a record within a structure occurrence. This allows the user to map an item of the external record from an item in an occurrence of a record within a structure (e.g., the first occurrence of an item within a structure).

6. Constraints

The constraint processing of geometry enforces geometry data integrity, which means that the system does not allow (1) ill-defined geometry to be stored and (2) changes to existing data that result in ill-defined geometry. Such control in existing systems is usually difficult. First, geometric entities are usually composed of multiple record types containing multiple record occurrences. Since it is not possible to store these entities with a single data base command, storing single data base records will leave the system with invalid geometry data. Second, leaving the enforcement of data integrity to the application user of the DBMS can result in the storage of invalid geometric data, since constraints exist as to which stored data items constitute valid geometric entities.

The constraints to be enforced during the DML operations defined above are defined in the schemas. These constraints are checked whenever a structure occurrence is about to be updated.

The majority of constraints used to ensure geometry data integrity are based on data associations defined in the schemas. Whether the DBA chooses to use CODASYL sets or foreign keys to describe associations, the schema language allows him to impose constraints between the records of the association. Other constraints may restrict the types of operations that can be performed on record types; e.g., the MEMBERSHIP DEPENDENT

constraint requires that an owner record exist before any member record is stored.

Another type of constraint, called a key constraint, restricts the values of items used to identify records. The two constraints used in geometry schemas are DUPLICATES NOT ALLOWED and NULL NOT ALLOWED.

The DBA may choose to restrict operations allowed on records of a schema so that unauthorized modifications may be prohibited. Such a restriction is necessary in cases where read access may be desirable on the individual records of a structure but where structure modification must always be performed as an indivisible operation.

7. DML Operations

Operations available to the user to access structures are the same as those available to the system as a whole: STORE, DELETE, MODIFY, FETCH, FIND, GET, COPY, and REMOVE. While the IPAD canonical geometry forms are always contained within a structure definition, this does not exclude nongeometry users from using these facilities to access, store, or retrieve data.

Operations that access structures do so by following their specified paths and by performing the actions dictated by the constraint options on the sets and on the operation itself.

8. DML Processing Directives

The schema contains options to control the processing of DML operations on records in the data base. One important directive is the SOURCE statement, in which changes to the owner (foreign key) record items are propagated to member (foreign) records. The DELETE MEMBERS IF DELETE OWNER directive activates the deletion of member records when their set owner is deleted. The COPY MEMBERS IF COPY OWNER directive calls for a copy of members when an owner is copied. The DELETE OWNER IF SET EMPTY directive provides for the deletion of owners when the last member record of the set is deleted.

9. User Features

Several features exist within the system to provide the user with a more user friendly way of storing geometry. First, the system has the ability to assign system-generated names to items used to show record associations; thus, the user does not have to give these names explicitly in DML commands that store the data.

Second, the DBA may choose to assign default values to items so that items not stored may be assigned values by the system.

GEOMETRY SCHEMA DEVELOPMENT

The mathematical representations of geometry entities to be used in IPAD geometry data base management have been presented. For a DBMS to control and provide access to these entities, the geometry data must be organized into schemas. This section describes the schemas developed for IPAD geometry and the capabilities for geometry manipulation provided by the IPIP DML.

The following levels of schemas were defined to support data definition and manipulation of geometric entities:

1. An internal schema to specify the storage organization of the data

2. A base-level logical schema to describe the defining data of the entities

3. Upper-level logical schemas to support record-at-a-time DML commands on the entities

The internal schema will consist of internal copies of the record types from the base-level logical schema. The mapping of structures to the internal level is not valid in the current implementation.

The Geometry Base-Level Schema

The defining values for each entity, as given by its canonical form, are contained in a base-level schema. Along with the mathematics to produce the shape, this information is sufficient to allow reconstruction of the entity.

The hierarchical relationship between entity types was used in organizing the defining data items for the entities into record types and structures; thus associations between entities are built into the schema. The associations are further specified using such DDL constructs as CODASYL sets (foreign keys) with constraint options, source declarations, and key constraints. The integrity of these associations is maintained by the system by enforcing the constraints during DML processing.

The entity types selected for this schema were OBJECT, COMPOSITE SURFACE, PATCH, COMPOSITE CURVE, SEGMENT, VERTEX, and VECTOR. Other entities may be added to the schema and associated with the ones already declared. Figure 2 showed the hierarchical

relationship between these entities. Each entity may be defined in terms of its component entities, as shown in the hierarchy. Thus, a segment is composed of its two end vertices, and a composite curve is composed of a list of its segments. These relationships are the foundation for organizing record types in the base-level schema.

We take the approach that an entity is stored by referencing its component entities. Thus, a segment is stored in terms of its two end vertices rather than the values for its end points and tangents. This will permit the sharing of vertices or points by segments. Thus, a modification of point values will affect all of the segments to which the points belong.

The base-level schema is declared by definition of record types, associations, constraints, structures, and DML directives.

Definition of Record Types.- The methodology used in defining record types for the geometry base-level schema is outlined below:

1. Data items are determined from the defining values of each entity, as given by its canonical form.

2. An identifier composed of one or more data items is determined for each entity type. Data items whose values are uniquely determined by the entity identifier are grouped into a record type with that identifier as a unique key. However, an item having several values for a given entity may belong in another record type with the identifier for the entity.

3. If a one-to-many association exists between an entity and its components, the identifier of the entity is placed in a record type for which the identifier of the component is a unique key. If a many-to-many association exists between entities, a separate record is used to model the association. The identifiers would comprise a two-item unique key.

These considerations will be elaborated and illustrated in the following description of the base-level schema for geometry.

Each of the defining values in the canonical forms of the entities is represented by a data item. For example, the position and tangent vector values in a segment vertex are represented by the item XYZ in the record VECTOR. The parameterization of a segment is represented by the item FLOW in the record type SEGLIST.

The entity name is used as the identifier for record occurrences which contain data items to define the entity. This identifier is a key item with the constraint DUPLICATES NOT

ALLOWED. The record types CCLIST, SEGMENT, and VERTLIST have unique keys CCID, SEGID, and VERTID and contain unique occurrences for each of the entities of types COMPOSITE CURVE, SEGMENT, and VERTEX, respectively. These records contain data items that are uniquely determined by the entity identifiers.

The association of an entity with its components is one-to-many and perhaps many-to-many, as previously described. We have chosen to allow a maximum flexibility of entity associations for IPAD geometry and, therefore, we have chosen a record configuration that allows many-to-many relationships between entities. This requires that the entity identifier and component identifer be items in some common record type.

A possible way to define record types to associate the entity with its component entities is to have one record for the entity type with items to identify the components. (See figure 3.) One occurrence of record SEGMENT would identify the two end vertices. This configuration has limited flexibility, since the maximum number of component entities is fixed by the number of record items. The number of vertex components needed varies with the type of vertex: one for LINEAR, two for CUBIC, three for QUINTIC. Thus, some items will be null for many record occurrences, or separate record types would be required for the vertex types. The addition of a vertex type that requires more derivatives and, hence, more components would require a change in the schema.

The number of associations (CODASYL sets or foreign keys) required depends upon the number of component entities; for example, two associations are required to link the SEGMENT record with the VERTEX record.

The schema that we actually constructed provides a configuration of record types for which the number of component entities may vary (fig. 4). The items END and COMP serve to order and thus identify the relationships of the components to the entity. For example, a segment has a vertex corresponding to the first flow parameter (END = 1) and a vertex corresponding to the second (END = 2). Each vertex in the example in figure 4 is CUBIC, but if a QUINTIC were required, a second derivative could be added as a component to the vertex by adding another occurrence in the VERTVECT record and with no change in the record declaration.

Only one association is required between record types with this configuration. There is, however, the disadvantage of more record occurrences, and the trade-off must be considered. Some of the record types and associations in the base-level schema are shown in figure 5.

Composite entities are defined by a list of primitive entities. Thus, a composite curve consists of a list of segments.

The record types to represent composite curves are CCLIST and CCSEG, as shown in figure 5. Here SEGID in CCSEG references a segment that is identified by SEGID in SEGLIST.

Associations.- As described in a previous section, the associations between record items may be explicitly defined using CODASYL sets or foreign keys. The arrows in the figures point from owner to member and constitute a one-to-many relationship. Thus, a particular composite curve has one or more segments associated with it via SET3, and a vertex may belong to one or more segments via SET6.

Constraints.- Constraints on the associations between entities serve to specify the exact relationship required by the entities. Enforcement of these constaints will ensure that only valid geometry is stored in the data base. Constraints may be added to these associations to specify allowable values for the number of members. For example, a segment must have exactly two vertices; a composite curve might be required to have at least one segment and, at most, 20 segments. A particular segment might be allowed to belong to no more than one composite curve. This constraint may be declared using the NUMBER specification in the CODASYL set or foreign key declaration. For example, SET5 should have NUMBER=2.

The constraint requiring that the two end vertices must exist if a segment is to exist is specified using the MEMBERSHIP DEPENDENT option on the set linking SEGVERT and VERTIST, SET6.

Structures.- The structures defined in the geometry base-level schema correspond to the entity types and contain records to be stored or retrieved together. Some structures we have defined are shown in figure 6. A segment to be retrieved may be identified by a value for SEGID in SEGLIST, which is the root of structure SEGMENT. Then all of the record occurrences related via the sets in the structure to the root occurrence for that segment will be retrieved. (See figure 4.)

DML Directives.- The data definition language constructs directing the processing through the multiple record structures representing geometry entities were described previously. Here we will show how they aid the manipulation of geometry entities and help to preserve the integrity of the data.

The integrity of data shared between record types may be maintained automatically by using the SOURCE construct described in a previous section. CCID in record CCSEG is declared as source from CCID in record CCLIST via SET3. A modifiction of CCID in

CCLIST will result in the change to CCID in CCSEG. However, a change only in the value of CCID in CCSEG would mean that the segment is to be associated with a different composite curve. Since SET3 is also MEMBERSHIP DEPENDENT, the curve identified by the new CCID value must exist in CCLIST.

The schema directives for multiple record processing are used to define the propagated DELETE, REMOVE, MODIFY, and COPY to allow update on an entire entity in one command. For example, the directive DELETE MEMBER IF DELETE OWNER on SET5 would cause the delete to be propagated from SEGLIST to SEGVERT in order to delete all the appropriate record occurrences for the segment being deleted. DELETE OWNER IF SET EMPTY on SET6 would cause the vertices to be deleted along with a segment unless MEMBERSHIP DEPENDENT prevents the deletion of a vertex used on another segment. Similarly, COPY MEMBER IF COPY OWNER will propagate the copy process from SEGLIST to SEGVERT via SET5.

Upper-Level Schemas and DML

Figure 7 shows some upper-level schema records developed to support IPIP DML on IPAD geometric entities. Each primitive geometric entity (e.g., VECTOR, VERTEX, SEGMENT, and PATCH) is represented in an upper-level schema record. The relationships of the entities that were defined in the base-level schema must be represented here as well. Their associations are shown by the CODASYL set (foreign key) definitions connecting the records in figure 7.

Each record for a primitive entity contains the items required to define and manipulate the entity. Because a record for a primitive entity contains all the defining data, only one DML command is required to access and update the entity.

Each of the composite entities is represented by two or more records. One record contains a unique occurrence for each entity of that type; the others associate the composite entity with its component entities in a many-to-many relationship. For example, a composite curve is represented by records CCURVE and CCSEG.

An upper schema could also be written to represent another type of geometry such as the ANSI communicaton standard. This geometry view could be mapped from the IPAD geometry base-level schema. These mappings require transformation between LaGrange and Hermite mathematical representations. The mapping schema would contain a definition of a many-to-many item transformation. For example, the four points representing a cubic curve in ANSI would transform to the two points and two tangents of an IPAD cubic curve. The next section will discuss the relationship of IPAD to ANSI geometry and schemas.

ANSI Standard in IPAD

IPAD supports the ANSI standard for communication of physical objects by means of a logical schema mapped from the IPAD base-level logical geometry schema.

The ANSI logical schema captures the ANSI standard as much as possible by using a record type for each ANSI structure type. (The ANSI structure type is not to be confused with the IPAD structure.) The actual conversion between the ANSI standard in the ANSI schema and the IPAD representation in the logical schema is specified in the mapping between the two schemas.

This approach leads to several complications. Most ANSI structure types can map to several IPAD geometry entities (e.g., the CUB structure type can be either a cubic curve or a cubic loft surface); conversely, an IPAD entity can map to one of several ANSI types (e.g., a bilinear patch could be mapped to the ANSI LIN structure or to the rigid body translation). To avoid confusion, each IPAD logical record type contains a field that is used to specify the ANSI structure type to which a record occurrence maps. When an entity is stored through some other schema, this field will be set to a predetermined value to indicate an ANSI structure type.

A second complication arises from the disparate way each logical schema defines geometric entities in terms of lower-level entities. Situations can arise where the modification of a value of an ANSI schema entity requires a cascading series of changes to values of other entities in order to retain integrity. To prevent such nonintuitive side effects, manipulation of ANSI external schema is limited to storage and retrieval commands only. This is not considered a severe restriction, since the ANSI standard is designed to be a communication standard, not a design standard.

Functional Capabilities of IPAD Geometry and IPIP DML

The functional capabilities of IPAD geometry in terms of the IPIP DML may be summarized:

STORE: Allows the creation of all entities, both primitive and composite, and the addition of entities to composite entities. Continuity constraints may be specified when appropriate.

MODIFY: Provides for the modification of any defining attribute of an entity. Names and values may be changed. Continuity constraints may be checked.

COPY: Creates a duplicate and disjoint entity.

DELETE: Deletes an entity from a composite entity.

REMOVE: Disassociates entities from a composite entity.

FETCH: Locates and retrieves entities.

FIND: Locates one or more entities.

GET: Retrieves the entity located in the most recent command other than a store.

These interpretations depend on the usage of the command with the appropriate record type. The record CCURVE, for example, is used with STORE when initiating a composite curve, e.g., when creating a composite curve with no segments. The record CCSEG is used with STORE to add segments to the composite curve. CCURVE would be used to DELETE or COPY a composite curve, while CCSEG would be used to REMOVE a segment from the curve or to FETCH a segment in the curve. A MODIFY on CCSEG could change the defining values of a segment or determine to which composite curve the segment belongs.

The following examples serve to illustrate the use of the DML commands on the IPAD upper-level schema to perform these geometry capabilities. A more complete and detailed description of the DML and its semantics and how they may be applied to geometry will appear in the IPAD user documentation.

1. To initiate a composite curve:

```
STORE CCID='CC1' IN RECORD CCURVE
```

2. To create a segment using previously defined vertices and add it to a composite curve:

```
STORE CCID='CC1',
     SEGID='SEG1',VERT1='VERTEX1',VERT2='VERTEX2' IN RECORD
     CCSEG
```

FLOW will be assigned the value (0,1) by default.

3. To extend a composite curve to contain a new segment whose second vertex is given by position and tangent values (fig. 8):

```
STORE CCID='CC1', SEGID='SEG2', VERT1='VERTEX2',
POS2='POINT3', PXYZ2=(4,6,-1),TXYZ2=(-1,1,0)
IN RECORD CCSEG
```

4. To add an existing segment to a composite curve and join it to a segment with C0 continuity (fig. 9):

```
STORE CCID='CC1',SEGID='SEGG3',END='1',TOSEGID='SEG2',TOEND='2'
IN RECORD CCSEG CONTINUITY C0
```

5. To change the value of an endpoint of a segment:

```
MODIFY PXYZ1=(2,3,7) IN RECORD SEGMENT
WHERE SEGID='SEG2'
```

6. To delete a segment:

```
DELETE RECORD SEGMENT
WHERE SEGID='SEG2'
```

The segment will be deleted unless it is in a composite curve or an object. Each vertex will be deleted unless it is in another segment or object.

7. To remove a segment from a composite curve:

```
REMOVE RECORD CCSEG
WHERE CCID='CC1' AND SEGID='SEG2'
```

The segment will not be deleted.

8. To copy a segment:

```
COPY SEGID='SEG4' IN RECORD SEGMENT

WHERE SEGID='SEG1'
```

The segment SEG4 will be stored with the flow, end position, and tangent values of SEG1 but with new identifiers for each vertex, position, and tangent vector.

9. To retrieve a segment in a composite curve:

```
FETCH FIRST RECORD CCSEG
WHERE CCID='CC1'
```

The first segment in CC1 will be retrieved.

CONCLUDING REMARKS

The work presented in this paper is the result of a team effort. IPAD's engineering group has supplied to the development team the requirements for managing geometry. The development team, including members from the DBMS, geometry, and scientific programming areas, has developed the strategies to manage the geometry data. One important factor in this solution is that the constructs, data structures, constraints, and mathematical procedures are defined through IPIP's data definition language. This allows each group using IPIP to customize the required data structures.

Schemas presented in this paper aim at supporting IPAD geometry. By adding more record types, all data necessary to support the geometry-related applications and other engineering tasks will be available from one source. With this approach, a truly integrated engineering data base can be achieved.

REFERENCES

1. NASA Contractor Report 2982, Product Manufacture Interactions With the Design Process (Boeing document D6-IPAD-70011-D), March 1979.

2. NASA Contractor Report 2981, Reference Design Process (Boeing document D6-IPAD-70010-D), March 1979.

3. Davis, P. J.: Interpolation and Approximation, Ginn-Blaisdell, 1963.

4. Date, C. J.: An Introduction to Data Base Systems, Addison-Wesley, 1977.

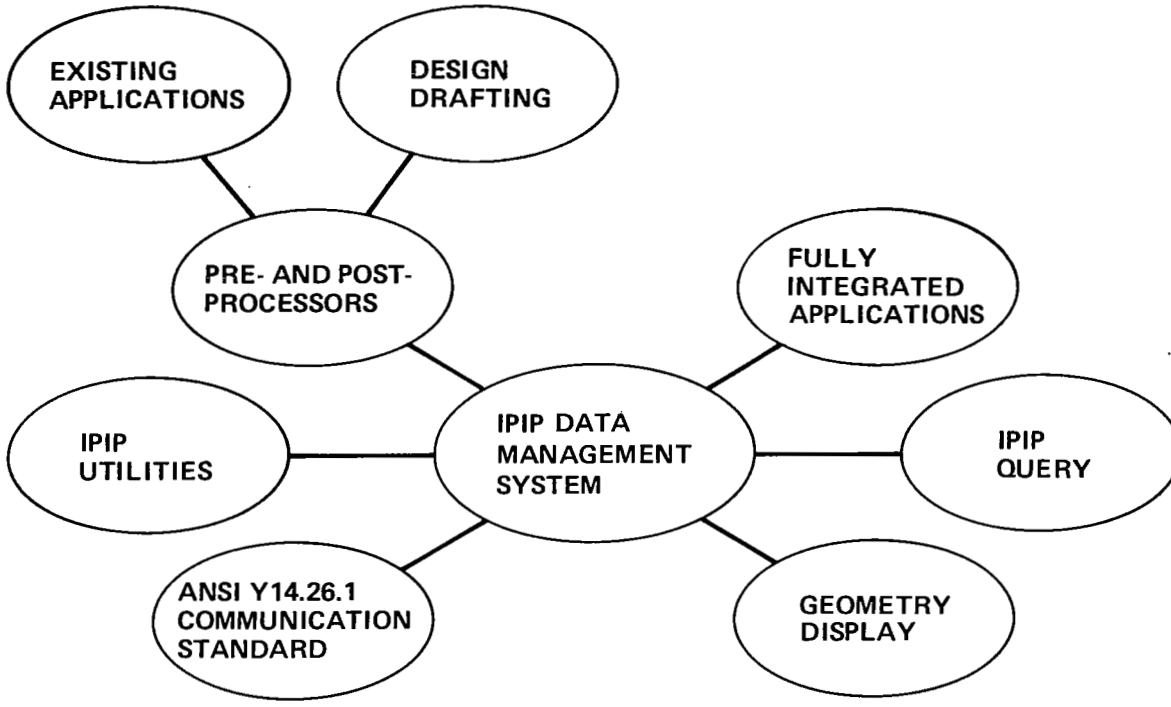


Figure 1.- Integrated data management.

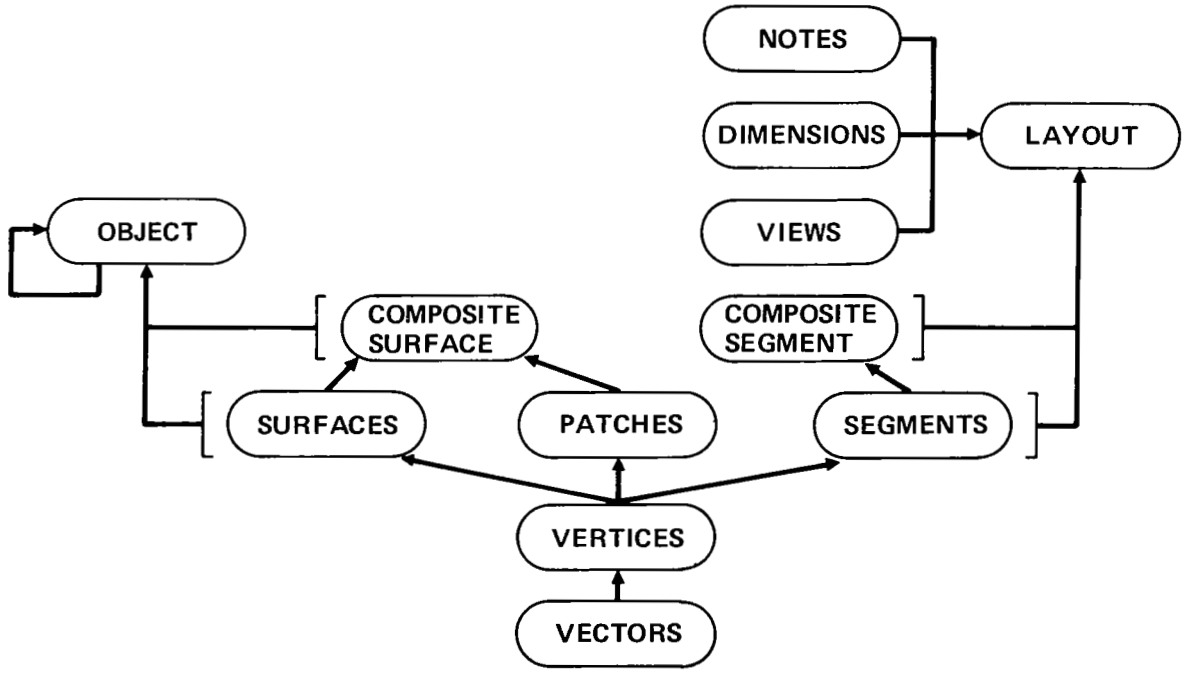


Figure 2.- Geometry hierarchy.

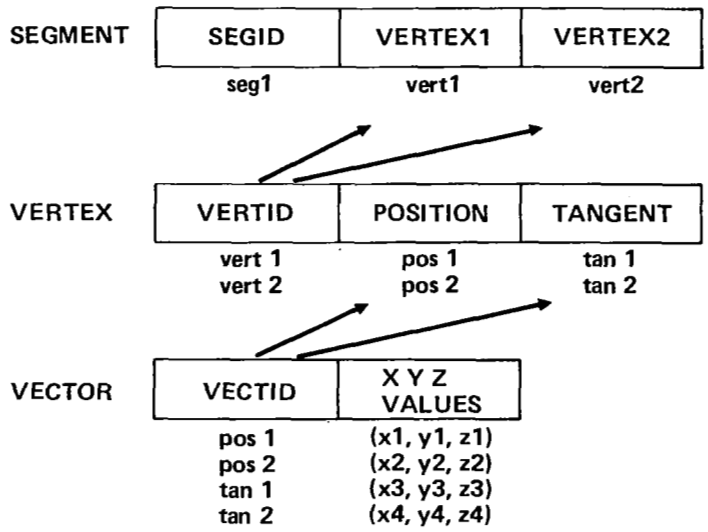


Figure 3.- Record configuration (number of component entities fixed.

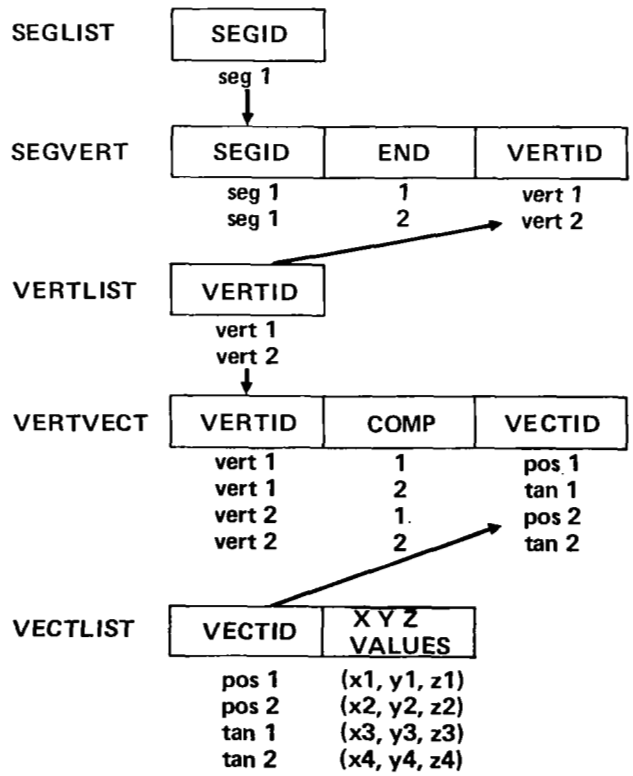


Figure 4.- Record configuration (number of component entities may vary.

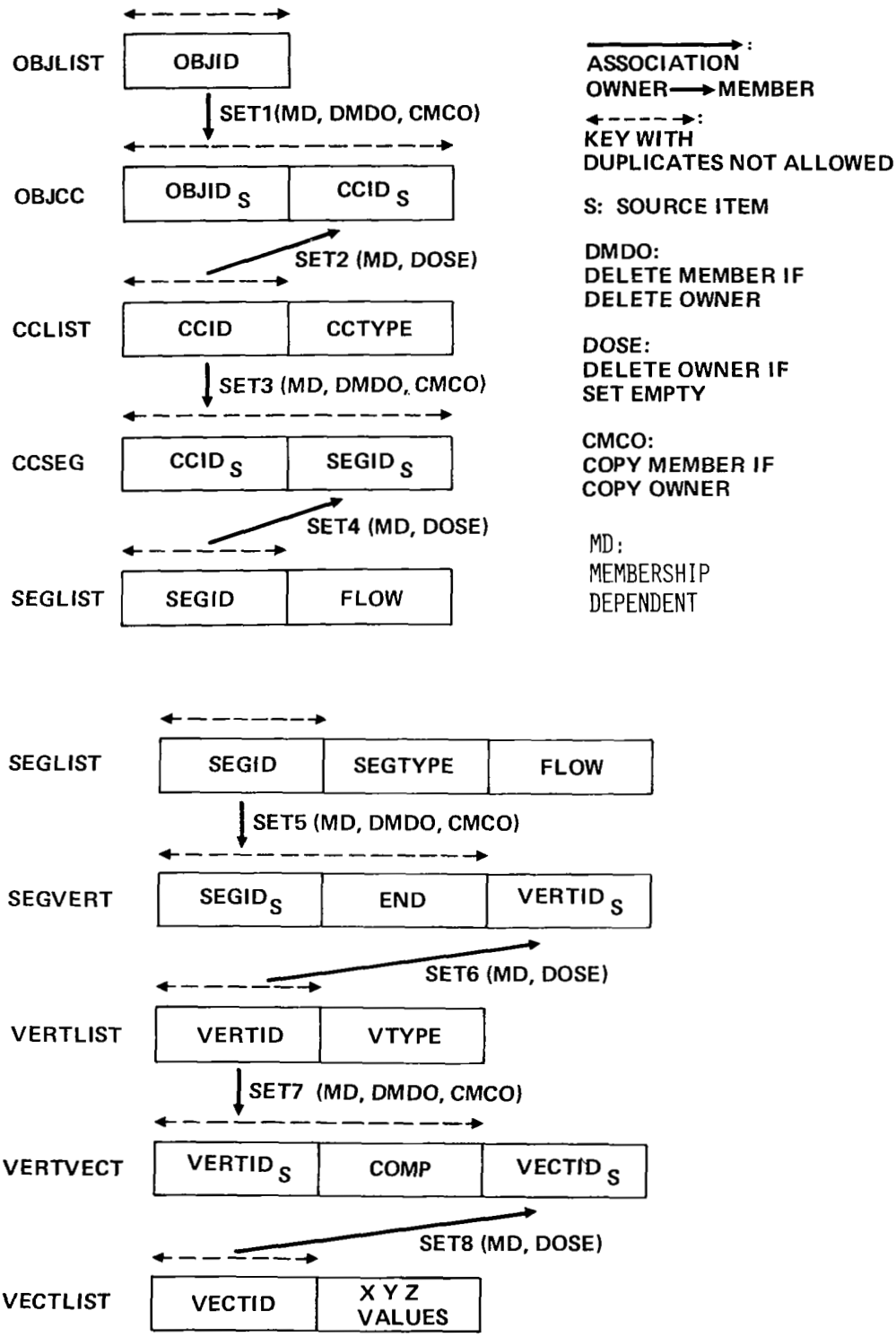


Figure 5.- Base-level schema.

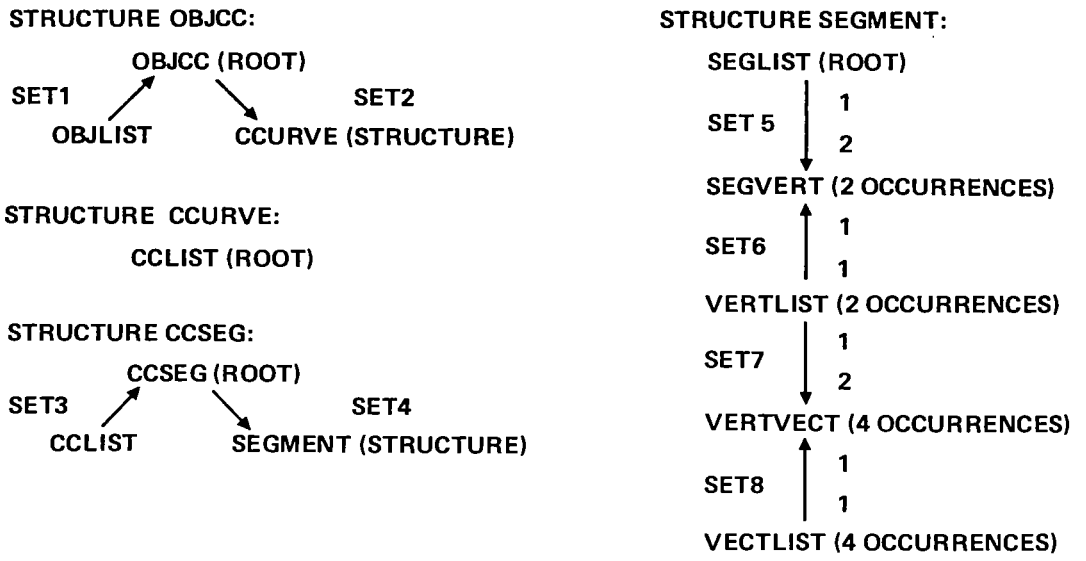


Figure 6.- Base-level schema structures.

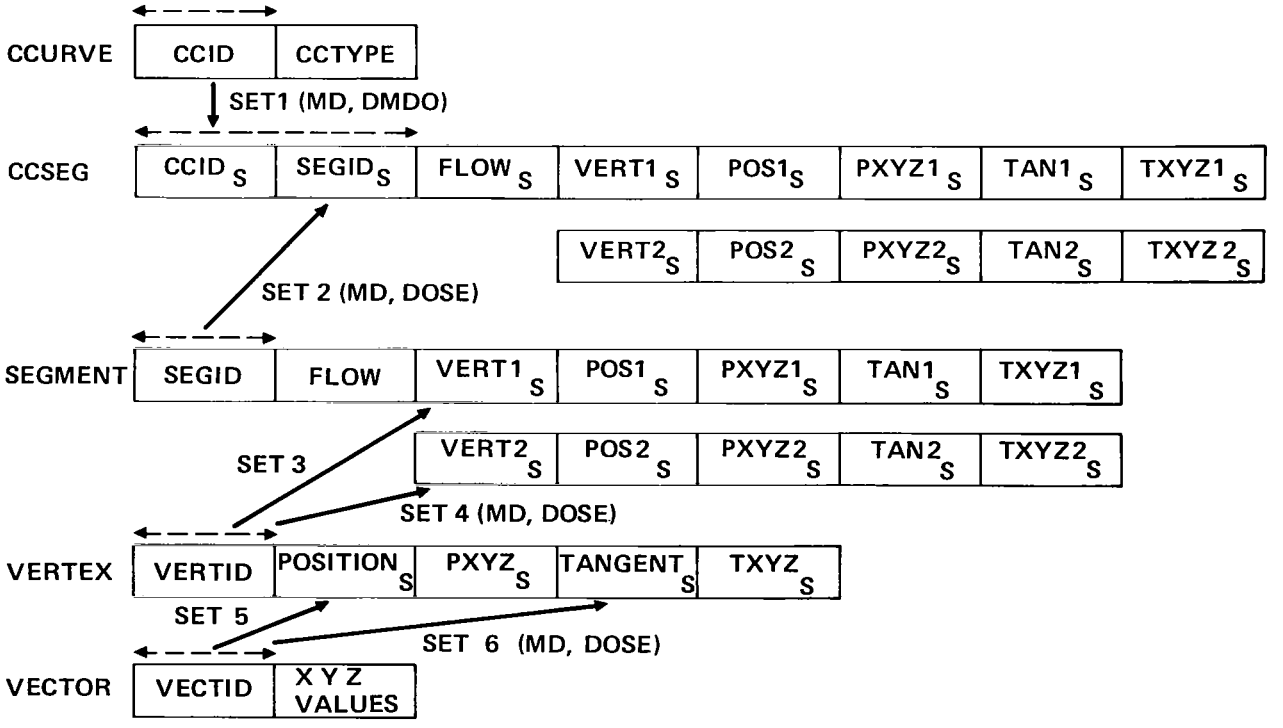


Figure 7.- Upper-level schema record types.

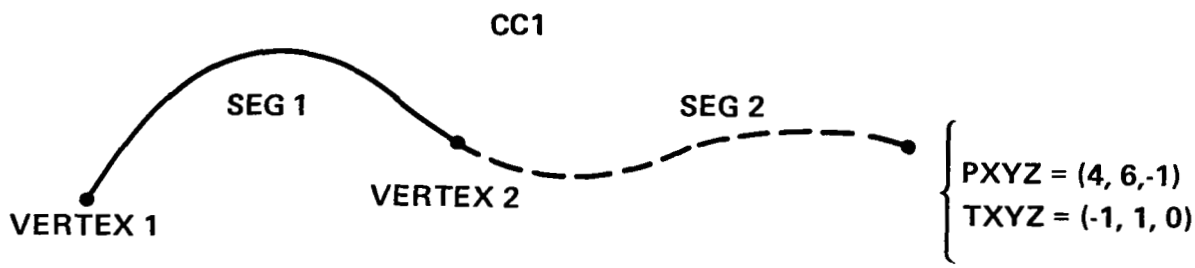


Figure 8.- Extending a composite curve.



Figure 9.- Segments joined with continuity.

USER INVOLVEMENT IN IPAD SOFTWARE DEVELOPMENT

Walter A. Bryant and Harold A. Crowell
The Boeing Company

SUMMARY

Since the contract go-ahead for the IPAD (Integrated Programs for Aerospace-Vehicle Design) system, there has been extensive and effective user involvement in support of the system requirements and software development. Through the selection and development of demonstration modules with supportive application programs, the user involvement continues to test and evaluate the prototype software. The feasibility of follow-on development can be established on the basis of the documentation and demonstrations in conjunction with informal information that will be available from the user representatives at the time of final contract delivery.

INTRODUCTION

The papers presented previously in this symposium have described the aerospace design environment and the associated information processes that must be supported by IPAD. System components have been described by those responsible for IPAD design. These discussions have addressed the requirements and capabilities of a "full" IPAD system, as well as some aspects of the prototype components.

This paper addresses the extensive user involvement in the software development of IPAD and the functionality of the IPAD prototype as viewed by the user. Although not a production system that can support an ongoing design process, the IPAD prototype is useful for the potential user as well as the interested system designer and is an essential tool for the companies committed to the use of the IPAD system. In this paper "user" refers to the engineer or manager responsible for the design, manufacture, or maintenance of a product, together with those supporting these functions. (Developers of software systems are users in another sense but are not included in this discussion.)

Representatives of the User Community

Immediately following the introduction, this paper describes the broad base of user representation that exists in support of

the development of the IPAD system. The engineering staff represents various functional backgrounds and has access to the experience of a broad spectrum of users through an advisory council, members and observers of the Industrial Technical Advisory Board (ITAB), and staff members of the IPAD Program Office (IPO) at NASA Langley.

User Involvement

The next section describes the roles and interactions of the various user groups, a process that remains much as it has existed from the beginning of the contract and is expected to continue, in some form, even beyond the software delivery. The daily interaction between the engineering and computing staffs becomes more rigorous as the time for software delivery approaches. The usefulness of the scenario as a communication tool is described.

User Evaluation of the Prototype

Finally, the paper evaluates the functionality of the IPAD prototype from a user's viewpoint. Demonstrating the prototype to the user community is the responsibility of the engineering staff. The demonstrations applied have evolved from the scenarios. These demonstrations and the application programs being developed to support the demonstration process are described. In support of the demonstration planning, a matrix was developed to map the prototype capabilities against the demonstration requirements. The matrix has also provided useful visibility of the prototype and the development that follows.

REPRESENTATIVES OF THE USER COMMUNITY

The interests of potential users of the IPAD system are being well served by a broad base of experienced representatives of the user community who have been involved since the contract go-ahead in April 1976. Results of earlier feasibility studies have also been incorporated, to the extent practical, to provide user information. Functional groups represented, as well as their applicable experience, are described below. Figure 1 depicts the user representatives and their relationships.

IPAD Engineering Staff

The IPAD engineering staff was organized at contract go-ahead and remained static in size and personnel for the first three years. It consisted of a manager and four engineers. The manager and one engineer were from structures; the others represented

preliminary design, structural design, and operations technology (computer-aided manufacturing). Of this staff, two members had been involved in the feasibility study (one full time), and three had been involved with the contract proposal. The initial effort required a temporary buildup of several engineers from weights technology, computer-aided design, manufacturing, tooling, and quality assurance, among others. Last year, the basic makeup of the group changed somewhat. The manager was named IPAD Assistant Program Manager, one of the engineers was made Engineering Manager, and another moved to a Boeing organization that was formed to implement IPAD. Subsequently, three engineers have joined the staff from within the company, one from structures and two from manufacturing engineering. An engineering aide completes the present staff. Whereas at contract go-ahead the engineering staff was close to the size of the computing staff, it is now outnumbered by nearly seven to one.

Boeing Advisory Council

An advisory council was organized early in the program to support the efforts of the engineering staff. The council was made up of Boeing managers and engineers from various organizations, including technical staffs, design projects, and manufacturing. The council was directly involved with IPAD during the first year, but as development progressed, its role became that of keeping abreast of the systems status.

Industrial Technical Advisory Board (ITAB)

ITAB was organized to support the User Involvement Plan (Boeing document D6-IPAD-70000-P). Its membership has been fairly static, consisting of 20 representatives of aerospace companies, engine manufacturers, and computer firms. The number of observers has varied, averaging 40 represenitives of aerospace and other industries, academic institutions, and Government agencies such as the Air Force ICAM program. Formal meetings occur on a quarterly basis, but other interactions can take place at any time.

NASA IPAD Program Office

The NASA-IPO staff monitors IPAD development, approves documentation and plans, and generally acts as the IPAD contracting agent. Acting as individuals with experience in engineering and computing science, IPO members contributed critiques and recommendations in the user community interests. The next section discusses how these various groups interact.

Consultants

Independent consulting firms have been placed on contract to support the IPAD development process.

USER INVOLVEMENT

The intensive user involvement in the software development of IPAD is best described by the interactions between the various user groups and by their interfaces with the IPAD computing staff. The following sections describe major program milestones and the user groups involved.

Definitions and Requirements

The initial period after contract go-ahead saw an intensive effort by the engineering staff to develop definitions and requirements. Defined were (1) the design process, (2) manufacturing interactions with the design process, and (3) management systems. Based on these definitions, the system and user requirements were developed. These were given to the advisory council in the forms of presentations and drafts for their critiques. Many of their recommendations were incorporated, and the revised documentation was issued to NASA-IPO for review and approval. Then followed an iteration to review and incorporate these comments as applicable. Concurrently ITAB was involved in the review process, first as an audit team, then as a full board review. This effort was in the form of five different documents, so the review process took place over several months. Figure 2 shows the evolution of the documentation.

The engineering staff was also involved with the development of the technical plan, management plan, documentation plan, and user acceptance plan during the initial period of the contract.

System Specifications and Preliminary Design

The computing staff developed a set of IPAD requirements based on the engineering definitions and requirements. The preliminary design of IPAD then got underway. The engineering staff now became the reviewers and critiqued computing documentation that described the various IPAD components. The preliminary design material was also monitored by NASA-IPO and presented at ITAB meetings. The culmination of this effort was reached at the critical design review in September 1978. This was combined with an ITAB meeting in Seattle. Engineering and computing presented various aspects of the program status. This

review resulted in the approval of preliminary design and a go-ahead for detail design and coding.

Prioritization of Requirements

In order to select a subset of requirements to be supported by the first-level prototype, the full set had to be prioritized. ITAB and NASA-IPO were called upon to support this effort. The feedback indicated the requirements determined to be mandatory, with the others weighted according to their needs.

The results of this exercise were used to prepare the IPAD User Requirements Implementation documents. Volume 1 of Boeing document D6-IPAD-70016-D-1 explains the requirements for first-level IPAD, volume 2 for second-level, and volume 3 for third-level. First-level IPAD is the prototype built for the current contract, while the others are intended for future follow-on contracts.

Requirements Update and Enrichment

The interaction between the engineering and computing staffs has continued throughout the program. One concern has been supporting the engineering interpretation and computing implementation of the requirements. A procedure called Interpretation and Implementation (I&I) is being used to control this interchange. The I&I forms become informal documentation based on either a request from computing for engineering support in the interpretation of a requirement or an input to computing by engineering when it is felt that engineering requirements are being overlooked or misinterpreted.

Formal changes to the requirements documentation are accomplished through the Configuration Change Board, in accordance with the Configuration Control Plan (Boeing document D6-IPAD-70005-P).

Scenario Process

Another informal process that evolved to meet the need for communication between the user and the computing staff is the scenario process, which is based on scenarios that depict various aspects of the design process and its interfaces. Using a modeling technique called SAMM (Systematic Activity Modelling Method), activities and data flow were selected from various design levels and presented to computing staff representatives each week. Computing would then respond in reference to IPAD support for the scenario based on full IPAD and the prototype. A new scenario package was presented each week, and the ensuing discussions clarified many points for both engineering and computing. This process provided the basis for detail design

changes and has provided the computing staff with a greater appreciation for engineering activities.

Figures 3 and 4 show the scenario contents and interactions with the ITAB applications subcommittee. This subcommittee was formed to work with IPAD on performance estimates for pre-IPAD engineering performance that will be compared to the performance using IPAD. The scenarios have been used for software design evaluation and demonstration development.

Information Surveys

Application Programs: ITAB and NASA were called upon to provide information in addition to the requirements exercise. In 1977 they were asked to identify application programs in the public domain as candidates for use with the prototype for demonstrations. They also provided other information concerned with the demonstrations that helped the engineering staff in its planning.

Geometry: In 1979 ITAB and NASA were requested to reply to a geometry questionnaire that asked how and what geometry was used in their constituent organizations. This information was used in support of the IPAD geometry software development and the IPAD geometry standard reseach that the engineering staff documented.

Demonstration Process

The demonstration process has included the selection and planning of demonstration packages from the scenarios that represent the design process and its interface with manufacturing. The demonstration package has been described to ITAB and NASA for their review. The demonstration package is described in the next section.

ITAB Visitations

Several ITAB companies have sent teams to visit the IPAD contractor to monitor progress and exchange ideas for IPAD development and implementation. The engineering staff participated in dialogues with these teams and realized benefits from these user exchanges.

Consultants

Notable among outside consultants that have contributed to the IPAD development is Information Research Associates.

Engineering scenarios have been used by this company to model the software design and measure performance of the IPAD system components.

USER EVALUATION OF THE PROTOTYPE

This section discusses the IPAD first-level prototype from the user's viewpoint. The discussion is based on the experience of the engineering staff in support of demonstration planning. The engineering staff is responsible for demonstrating the prototype to the user community. In addition to describing the use of the prototype in conjunction with a demonstration package, the section generally describes how full IPAD utilization would differ from the prototype experience.

Demonstration Planning

Engineering scenarios were developed with the dual purposes of communicating engineering needs to the computing staff and providing a basis for demonstration planning. Excerpts were selected from the scenarios to provide a continuous subset of activities that would begin in preliminary design and proceed through detail design to the interfaces with manufacturing. This would form a demonstration package made up of modules that can be demonstrated as individual sessions at a terminal. The criteria for selecting demonstrations are shown in figure 5.

To provide visibility for demonstrating planning, a matrix has been developed to map the IPAD prototype capabilities against the demonstration requirements. The matrix is also useful as a picture of the prototype and as a base for planning follow-on IPAD levels. The matrix indicates the requirements to be supported by the prototype (level I) and follow-on (levels II and III) stages. The demonstrations are mapped against these, indicating requirements used by each. The prototype capabilities are mapped against these requirements and the phased releases are reflected. If demonstration requirements are not supported by prototype capabilities, the need for an application program is indicated.

Application Programs

In order to support the demonstration effort, computing has established an application programming group. This group is responsible for evaluating the engineering specificaton of each demonstration for IPAD capabilities and for providing a computing interpretation of the specification along with resource flow times and milestones. In each demonstration, applications programs will have to be written or, in some cases, converted to permit the use

of IPIP. There are two ways to integrate applications programs into IPAD, one is fully integrated and the other interfaced. Fully integrated programs make direct calls to IPIP (IPAD information processor), where the interfaced applications require pre- and postprocessors to take advantage of IPIP capabilities. Both aspects of integration will be shown in the demonstration package.

To develop the best possible demonstration package, a procedure was established for design reviews and walkthroughs as each demonstration is developed. A diagram depicting this activity and the current status of each demonstration is indicated in figure 6.

Demonstration Package

Demonstration modules to be developed and attendant application programs have been scheduled and numbered in a sequence based on the availability of prototype capabilities. The modules in the sequence to be used during the demonstration are described below and illustrated in figure 7.

Step 1. Project Management (Demo Module 3): This demonstration provides work breakdown structure, task and subtask definitions, and associated management data such as cost and schedule data, loads the data base with IPAD actual project data, and provides management reports through interactive query of the data base.

Step 2. Configuration Parameters (Demo Module 1): Parameters in the data base are initialized to characterize the airplane body, wing, empennage, power plants, and landing gear. The data is used by the configuration designer to develop a specific configuration.

Step 3. General Arrangement (Demo Module 2): The configuration designer develops discrete characteristics of the configuration. A wire diagram representing the general arrangement of the airplane is created.

Step 4. Structural Arrangement (Demo Module 6): The 3-D geometric model of the general arrangement is expanded using geometric entities to develop the structural arrangement, again represented by a wire diagram. A general structural model is created for use by staffs (structures, weights, etc.).

Step 5. Structural Analysis (Demo Module 7): The general structural model is interrogated to construct a finite-element (FE) analysis input file. The output of the FE analysis is then reviewed through a formatted presentation.

Step 6. Frame Design (Demo Module 8): Using frame contours from Module 6 and internal and external loads from Module 7, the designer reviews and evaluates the structural concept by determining whether the concept will adequately handle the loads developed in Module 7, then creates a detailed design of a body frame.

Step 7. Manufacturing Interaction--Indentured Parts List (Demo Module 4): Using the frame drawing from Module 8, the demonstration will interactively accomplish two main things: (1) create an IPL and (2) interrogate the PL/IPL. Other capabilities that will be demonstrated are:

Graphic IPL format
Error messages and tutorial
Record types for PL data
Interrupt capability including "skip-through" menus
English-like menu for PL/IPL interrogation

Step 8. Manufacturing Interaction--Geometry (Demo Module 5): This module will show that the geometry, as it exists in the data base, can be configured to meet manufacturing requirements through: (1) multiple views, (2) direct transfers of geometry to apt source statements (with preprocessor to AD2000), and (3) the geometry/parts list relationships.

Step 9. Data Base Administration (Demo Module 9): This demonstration includes miscellaneous data base administrator activities based on IPAD requirements.

Integrated Demonstration Package: The complete demonstration is processed with the final software release.

The total package above represents a flow of data through the design process and its interface with manufacturing. (See figure 7). The package will demonstrate the many IPAD capabilities in an integrated environment. Since the demonstrations are to exercise the IPAD capabilities and not to provide sophisticated production

systems, basic geometry and subsets of other types of data were used. Thus, this package demonstrates the data integration capabilities of IPAD in the engineering environment.

Using The IPAD First-Level Prototype

The majority of the user interface with respect to the demonstration package will be through each application program. This is representative of any prototype application; however several exceptions will be discussed in this paper.

Loading Data into IPIP.- As with any other data base management system, different data-loading techniques are available. The IPAD information processor (IPIP) is an engineering data base management system and therefore must serve the needs of the engineering community. This includes scientific data storage (which can be voluminous at times), geometric data descriptions requiring data structure to be associated with each entity type, manufacturing information, and accounting and managerial data. IPIP is being developed to support all facets of engineering. Because of this diversity in processing, a general-user data manipulation language (DML) has been developed that will support these engineering needs. The description of the DML can be found in the report describing the impact on application programs using IPIP, which is available upon request.

There are two ways of loading data into the IPIP data base: (1) through the query processor, and (2) through an application program. There will be logical schemas associated with each of these that describe the user's view of the data.

Manipulating Data.- There are two methods of manipulating data: through application programs and by QUERY processor; both use the data manipulation language (DML) for IPIP. When accessing data via applications, the user can be guided through record relationships of the data base without extensive knowledge of the DML or the data base. However, once a user becomes familiar with IPIP DML and related concepts he will be able to traverse the data base freely and will thus enjoy greater flexibility in making data inquiries. A sample query session is shown in figure 8.

Creating Geometry.- There are three ways to create geometry in IPAD: (1) AD-2000, (2) geometry display utility, and (3) application programs.

The primary source of interactive geometry construction is AD-2000, a computer-aided design drafting system that has been interfaced with IPIP by a pre- and postprocessor. The preprocessor allows access of IPAD geometry from IPIP for use in the drafting system, and, conversely, the postprocessor converts

geometry created in AD-2000 to the IPAD canonical forms for storage in IPIP.

The geometry display utility has been developed to allow users to combine existing geometry to create new geometry configurations and to provide a general viewing capability.

Application programs use general purpose graphics system (GPGS) calls for display of geometry, both as it is generated in a construction mode and after retrieval from the data base. IPAD also supports ANSI Y14.26.1 geometry formats for communication.

User Interface.- The prototype presents constraints to the user that full IPAD will not. For example, in the creation and use of geometry to support the demonstration, while the initial geometry for the general arrangement is developed with the application program, the structural arrangements and detail frame design are accomplished using AD-2000. The drafting system is on the DEC VAX 11/780, while the application programs are on the CYBER 170/720. The creator of geometry using AD-2000 must file his work, then log off the VAX and log onto the CYBER and run the postprocessor to convert and store the geometry. Conversely, to access IPAD geometry for use with AD-2000, he must run the preprocessor on the CYBER, send the converted geometry to be filed on the VAX, and then get on the VAX to use AD-2000. On full IPAD, these would be direct calls on an integrated computer-aided design (CAD) system. During actual demonstrations, other differences will be apparent to viewers.

As IPAD is enriched, the pre- and postprocessors will become integral parts of AD-2000 as imbedded applications that make direct calls to IPIP through the communication control program (CCP) facility available for high-speed network communications.

Prototype Testing and User Acceptance

The computing staff has the responsibility to integrate and test the prototype components for response time, throughput, usability, extendability, etc. By the time the engineering demonstration packages are run, this type of testing will be complete. The actual testing by the user starts in the scenario presentation phase and is concluded when the demonstration documentation has been completed and delivered.

Computing staff personnel also establish acceptance criteria for their testing. The user establishes acceptance criteria in the comparison of actual performance with estimated performance values. This is but one measure. He will also compare the IPAD prototype with other systems he is familiar with for parameters such as response time, user interface, etc. The final user acceptance will be by the total user community based on the

demonstration and other documentation (and preferably with some hands-on testing), but it must also be based on what an expanded usable system, employing prototype design concepts, can be.

CONCLUDING REMARKS

This paper can be summarized by reemphasizing the user involvement in the development of the IPAD software:

User involvement has been extensive and effective throughout the IPAD program. The design process has been the support objective and there have been many checks to ensure that the objective is met.

The scenario process has proven effective for communications and as a check to ensure that development has been on target. It should have been introduced earlier, during the requirements definition phase, for a stronger impact.

The demonstration package, as designed and developed to be run on the prototype software, illustrates the intent of a full IPAD system. The design concepts have been established and can now be projected. Thus, the matrix reflects a program foundation that goes beyond the prototype and encompasses the full IPAD concept.

The feasibility of follow-on development can be established on the basis of the demonstration package and by documentation in conjunction with the information released throughout the development program.

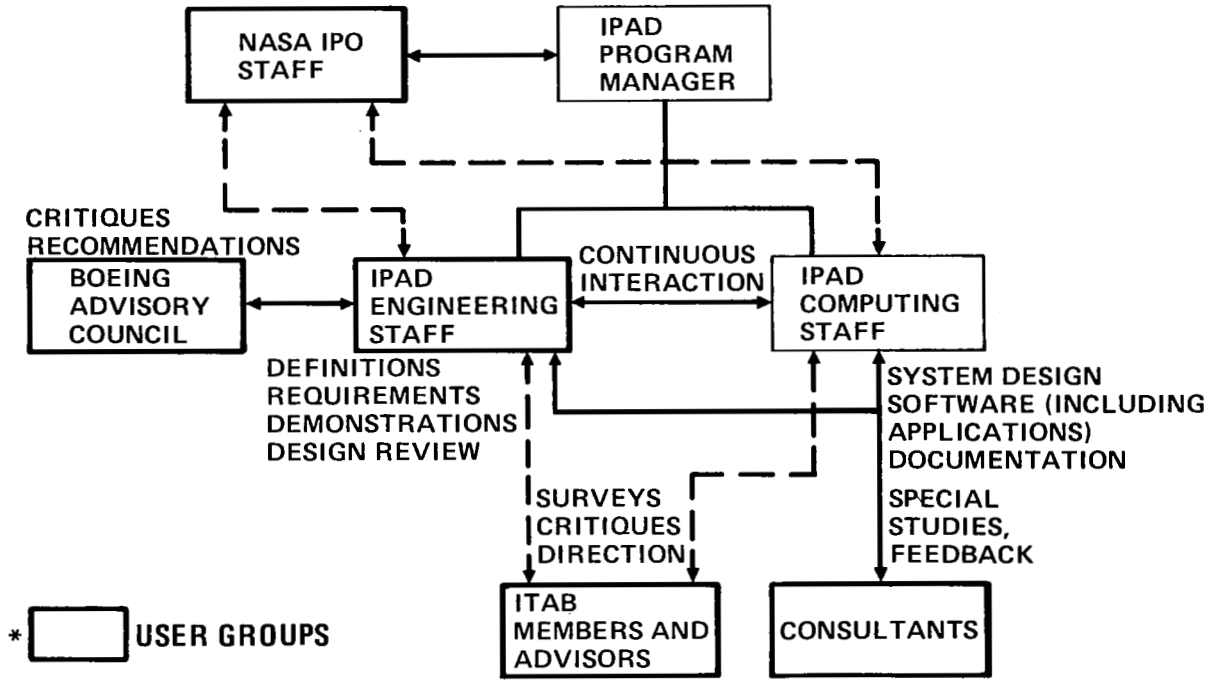


Figure 1.- User representative relationships.

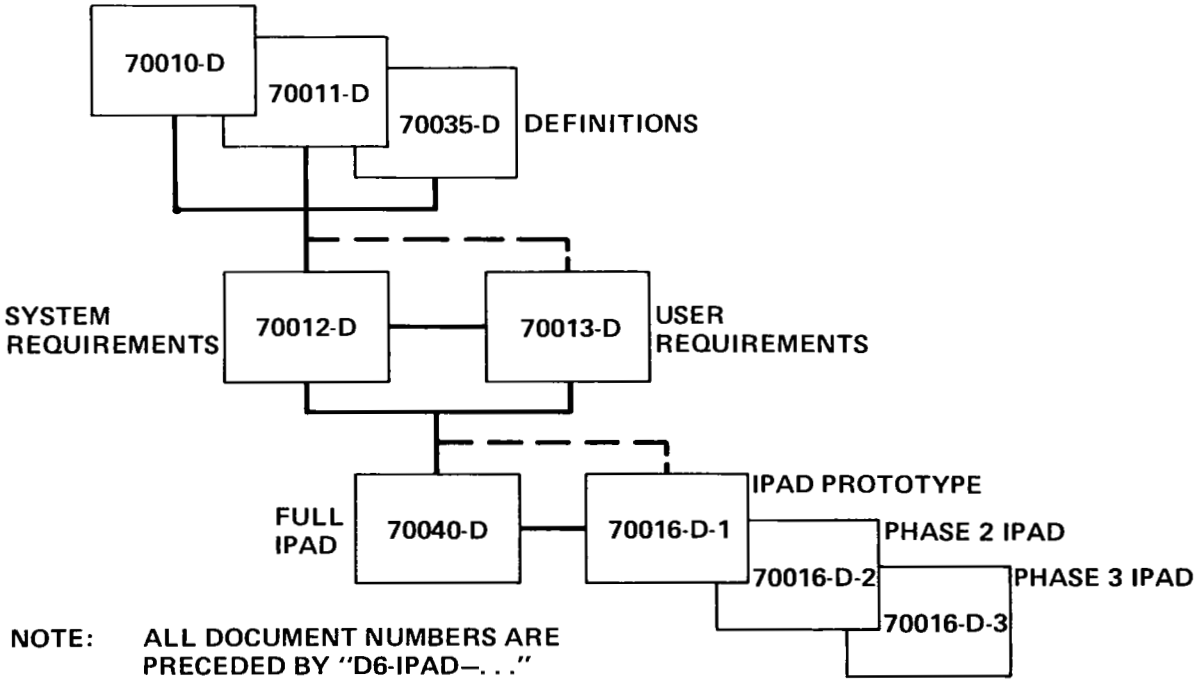


Figure 2.- IPAD definitions and requirements documentation.

- INTRODUCTION
- COMPARISON OF IPAD VS PRE-IPAD ENVIRONMENT*
- IDENTIFY INFORMATION AND ACTIVITIES
- DETAIL DATA BREAKDOWN
- DETAIL ACTIVITY BREAKDOWN
- IPAD IMPACT ON ENGINEERING ACTIVITIES
- IMPACT ON REQUIREMENTS
- SUITABILITY FOR DEMONSTRATION*
- IPAD EFFECTIVENESS ESTIMATES*

*ITAB APPLICATION SUBCOMMITTEE REVIEW

Figure 3.- Scenario documentation.

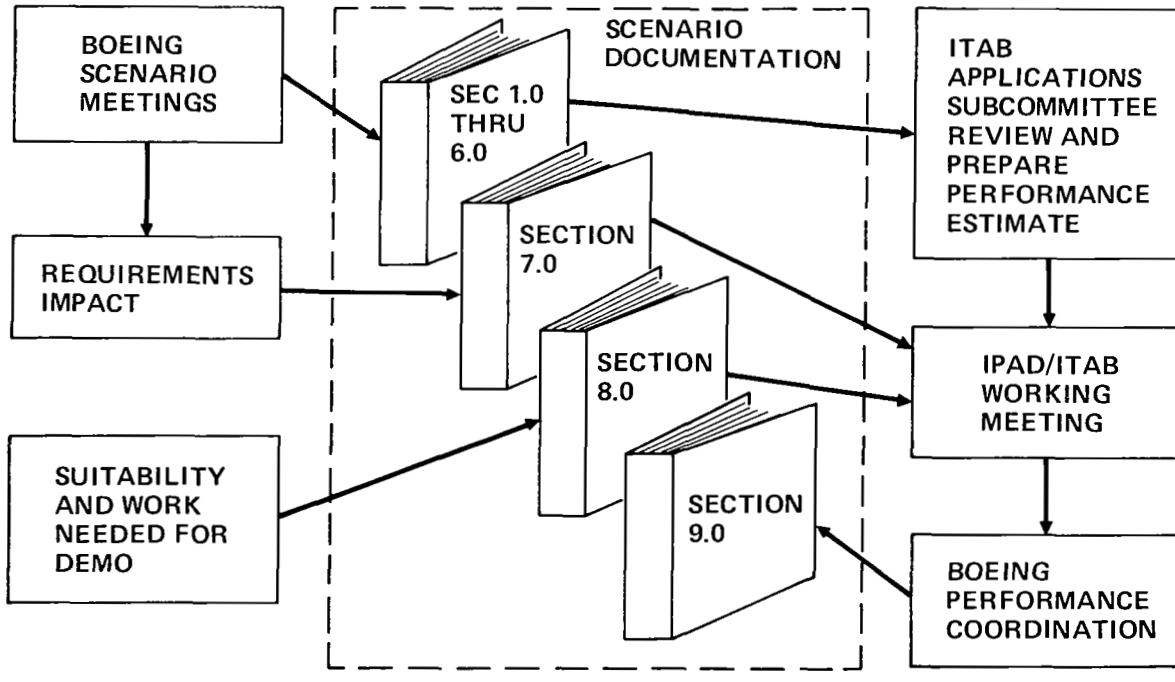


Figure 4.- Interactions with scenarios.

- SELECT FROM SCENARIO PROCESS
- DEMONSTRATE PROTOTYPE AND ILLUSTRATE INTENT OF FULL IPAD
- EMPHASIZE DATA MANAGEMENT, DATA COMMUNICATION, GEOMETRY
- CONSIDER FOR ENGINEERING DATA MANAGEMENT SYSTEM:
 - SUPPORT FOR VARIOUS DATA TYPES
 - AVAILABILITY OF PROGRAMS AND DATA
 - EMPHASIS ON REQUIRED FEATURES NOT AVAILABLE IN OFF-THE-SHELF SYSTEMS
 - MANAGEMENT OF DATA BETWEEN ENGINEERING COMPUTER PROGRAMS
 - INTERFACE BETWEEN IPIP AND AD–2000 DRAFTING SYSTEM
- CONSIDER FOR DATA COMMUNICATION
 - USE OF ENGINEERING DATA STORAGE TO SUPPORT ENGINEERING DISCIPLINES AND ANALYSIS THAT RUN ON THE CYBER AND DEC COMPUTERS AND ACCESS THE CENTRAL DATA STORAGE
- CONSIDER FOR GEOMETRY CAPABILITY
 - TRANSFER OF GEOMETRY INFORMATION BETWEEN COMPUTERS
 - TRANSFER OF GEOMETRY INFORMATION AMONG COMPANIES

Figure 5.- Criteria for selection of demonstration.

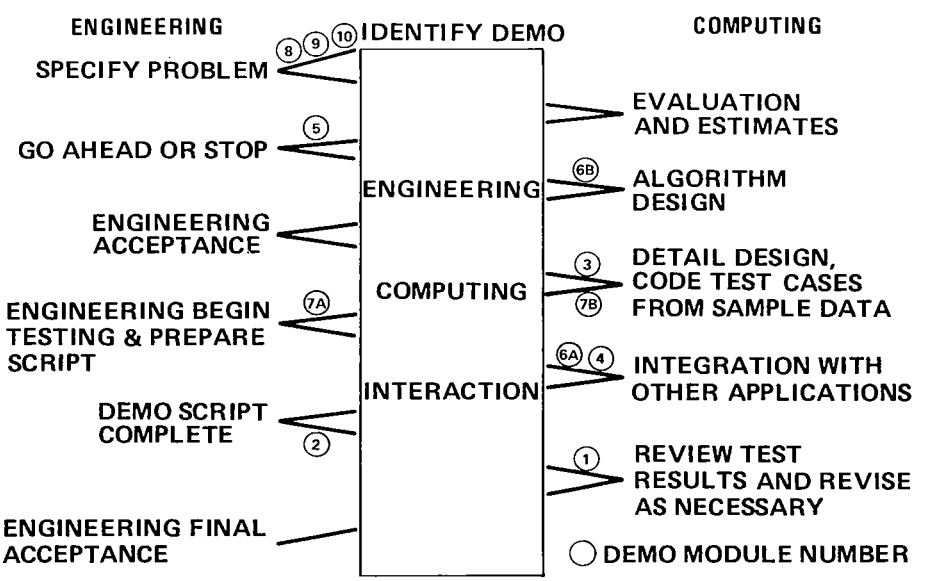


Figure 6.- Demo development procedure and status.

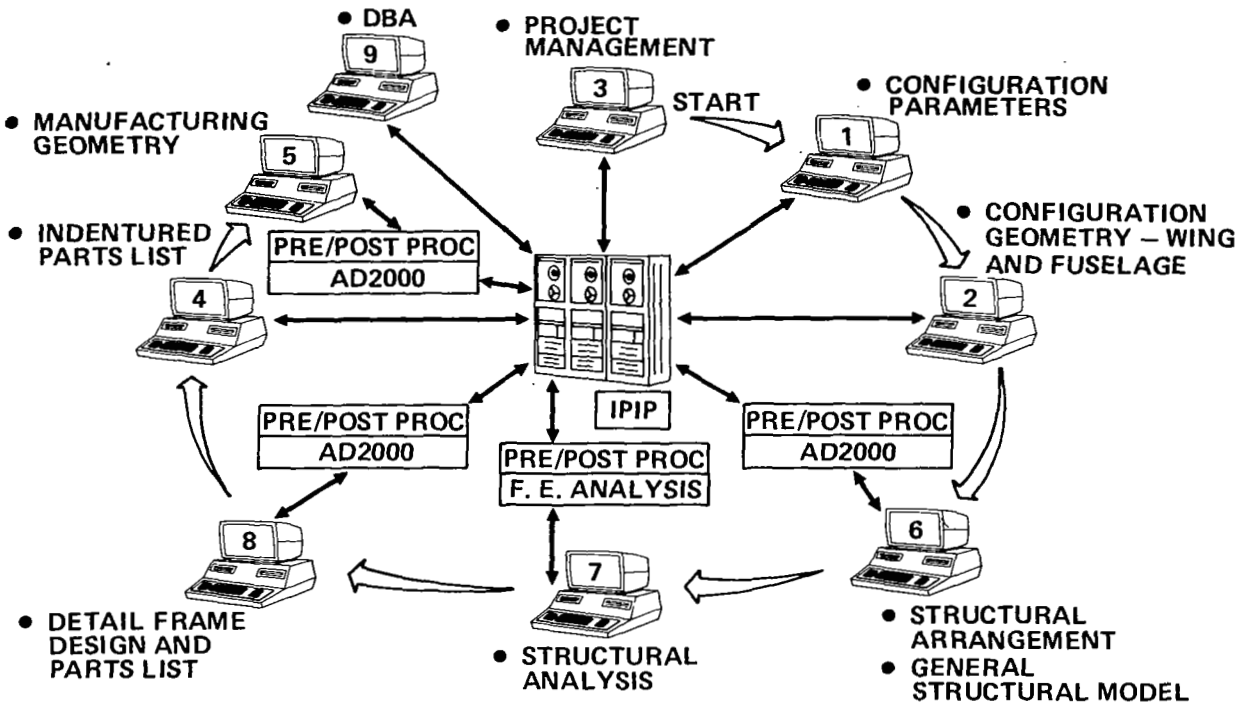


Figure 7.- First level integrated demo.

- OPEN SCHEMA=CONFIG
- ACCESS DATA SETS EDCFOOIA, EDCFOO2A FOR UPDATE
- DISPLAY ALL PFRMBT
- STORE MN=737, CN=WING, CNU=TEST1, AR=9.0, TR=0.3, SA=25.0 IN RECORD=PFRMBT WITH DATA SET =EDCFOO3A
- CLOSE SCHEMA=CONFIG

Figure 8.- Query processor loading.

IPAD PRODUCTS AND IMPLICATIONS FOR THE FUTURE

Ralph E. Miller, Jr.
Boeing Commercial Airplane Company

SUMMARY

Although the IPAD system takes its name from the aerospace industry where it was developed, it is actually directed toward the universal goal of human problem solving. This presentation is concerned with the solution of design and manufacturing problems. It specifically addresses the betterment of productivity through the improvement of product quality and the reduction of cost. Productivity improvement is sought through (1) reduction of required resources, (2) improved task results through the management of such saved resources, (3) reduced downstream costs through manufacturing-oriented engineering, and (4) lowered risks in the making of product design decisions.

It is the view of this presentation that productivity has reached a plateau due to the stagnation of computer utilization in the current product design and manufacturing processes. The hypothesis is formulated that the quest for new and higher productivity plateaus can only be achieved by large scale information integration. Large scale information integration needs new software such as the IPAD system and its products. These IPAD products are the consequence of solving five key problems relating to the integration of information with production activities: (1) very large numbers of human participants and specialized tasks directed toward a common objective (the "lots of" problem), (2) collection of pertinent information in distributed or centralized data bases, (3) data sharing and communication capabilities, (4) essential standards for effective communication, and (5) integration of user application programs.

The IPAD products are both hardware architecture and software distributed over a number of heterogeneous computers in this architecture. These IPAD products are described in terms of capability and engineering usefulness.

The future implications of state-of-the-art IPAD hardware and software architectures are discussed in terms of their impact on the functions and on structures of organizations concerned with creating products.

IPAD OBJECTIVES

The IPAD products have evolved from examination and analysis of the problem-solving environment, which consists of (1) humans joined in a common endeavor, (2) their computers, (3) information essential to their endeavor (data), and (4) the problems they must solve. To perform such an examination and analysis, it was necessary to determine the roles, relationships, and processes of all elements making up the environment and to seek ways to improve the problem-solving process.

During the evolution of product data, the prime objectives of its developers are to improve the quality of the product and control its cost. Figure 1 shows the two levels involved in the creation of product data. First, at the idea level, the primary concern is with conceptual events: preliminary design, detailed design and analysis, and product definition. Second, at the product level, the concern is with production processes: the release of parts, the planning of manufacturing operations, the development of tooling, and the manufacturing event itself. Throughout both of these levels, many participants are involved, thousands of labor hours are consumed, and constant review by management is required. In anticipation of this massive organized effort, the objective of IPAD is to facilitate the activities of each level and to provide for a powerful interaction between the two levels to the end that improvement in product quality and cost control will result.

PRODUCTIVITY IMPROVEMENT

The basic resources for problem solving are cost and flowtime, and the consequences of applying these resources are evidenced by results from the tasks to which they are applied. Productivity improvement can be manifested as (1) a reduction in one or both of these resources with no change in task results, (2) an improvement in task results with no change in resource requirement, or (3) a combination of these two benefits, figure 2.

The primary method of improving productivity is through reduction in required resources. The introduction of new methods results in less expenditure of cost and/or flowtime when performing a given task. Benefits from the use of new methods are as illustrated by the examples in figures 3 and 4 (ref. 1).

The second form of productivity improvement is achieved by managing the utilization of some of the resources saved by the use of new methods. Figure 5 shows that, by re-investing some of the saved resources in a carefully disciplined and managed process,

new plateaus of improved task results can be attained, and an improved level of productivity can also be obtained.

The third area of productivity improvement is reducing downstream costs. Figure 6 illustrates that, in a typical product development activity, a relatively small change during the engineering design can cause a magnified effect on the subsequent manufacturing operation. Thus, new methods that permit rapid engineering evaluation of design changes can reduce downstream operations--particularly out-of-sequence events in manufacturing--and thus enormously reduce total production costs. Figure 7 shows some examples of the potential for a typical 300-airplane subsonic transport program.

The final area of opportunity for productivity improvement is in the reduction of corporate risks. Figure 8 illustrates the impact of decision making on program cost, with preliminary design highlighted as the critical phase in this process. By the end of this phase even though only four percent of the total product cost has been incurred, about 70 percent of the decisions have been made; thus it will be seen that preliminary design represents a period of tremendous risk. The introduction of new methods and tools which optimize the quality of the engineering work performed through preliminary design reduces the risks inherent in the commitments made to this point.

PRODUCTIVITY STAGNATION

Even though nationwide efforts toward improved productivity have diminished considerably over the past 20 years, computers have contributed to notable successes in some industries (ref. 1, 2, and 3). However, many of these efforts have concentrated on specific tasks within a larger problem-solving environment and thus have exerted only limited influence on productivity in the broad sense. To illustrate this point, figure 9 shows relative amounts of computer program development at Boeing in the application areas of (1) technical analysis, (2) design, and (3) business systems. Shaded portions of each bar indicate the proportion of these computer programs that are integrated for optimum utility throughout the problem-solving environment. Despite 25 years of experience and effort with program integration at Boeing, broad utilization of this information integration technique is still very low. This small degree to which information is integrated within problem solving processes is evident fairly generally throughout American industry and is one of the primary causes of productivity stagnation.

NEW PRODUCTIVITY PLATEAUS

"Hard" Versus "Soft" Information

To date the search for productivity has concentrated on individuals, their tasks, and certain crucial physical work space and equipment considerations. These may be thought of as the "hard" information elements of the problem-solving environment for they are visible, tangible, and concrete. They possess simple data characteristics such as geometry topology, quantity, location, etc. They are, therefore, easily managed.

In contrast, there are certain "soft" information elements that are characterized by abstract qualitities; they are invisible, theoretical, and complex. These soft information elements must be represented by equations, mathematical symbols, and logic. Soft information is visible abstractly in drawings, analyses, computer printouts, and electronic displays. Obviously, it is vastly more difficult to manage soft than hard information, especially because of its volume and complex computer structures and relationships. As a consequence, there are few in-place computer programs that deal adequately with soft information.

Although it is understandable--as well as demonstrable--that industry has concentrated up to now on dealing effectively with hard information, it is essential, if we are to achieve new productivity plateaus in the future through large-scale information integration, that we also concentrate on handling soft data.

Data Base and Work Relationships

In the complex process of integrating information and work performance, the key entities are the product organization, its tasks, and the data base. The most crucial challenge is the construction of an effective interrelationship between these entities. Figure 10 shows this relationship and makes the particular point that both hard and soft information must be integrated on a total organizational basis. The IPAD system recognizes that it must be concerned not only with the specialized information needs of individuals and subgroups but with all of the various participating functional organizations that contribute to product creation. The common resource for implementing the relationship between work and information is the data base, and it is through this repository that all of the diverse activities--construction, display, execution, modification, and disposition of work--have common access to required information.

Action and Information

In evaluating total organizations and their tasks, IPAD is obliged to provide an environment where action can be integrated with information, for these are the basic elements by which human participants must be represented. Figure 11 shows that the user is found at all levels of activity within an organizational structure. Questions of "what" pertaining to that user's activities are primarily concerned with information in the organization, whereas questions of "how" relating to the organization's activities are essentially concerned with the actions of the user.

Figure 12 lists the five key problems to be solved if successful integration of action with information is to be achieved: (1) large numbers of people and specialized tasks (the "lots of" problem); (2) data bases, both distributed and unified; (3) data sharing and communication capabilities; (3) communication standards; and (5) integration of user application programs.

IPAD PRODUCTS

One unique accomplishment of the IPAD program has been the establishment of generic hardware and software architectures that address these key integration problems. As a consequence, these architectures provide for (1) multiple computer systems to ensure adequate capacity and reliability; (2) heterogeneous computer manufacturers and vendors to provide cost-effective matching of tasks with computer devices as well as competitive procurement; and (3) a hierarchy of communication capability, with special emphasis on channel speed levels to ensure effective data- and program-sharing.

As a consequence of this approach, the IPAD computer laboratories have been developing three different software implementations (fig. 13) for three different manufacturer's computers. The three computers are connected together by high-speed communications, thus providing megabit speed communication between these heterogeneous machines. Figures 14 and 15 show the detailed architecture of these software implementations. In these diagrams, a high degree of symmetry is apparent in the software capability existing across the three machines, thus ensuring integrity of user data regardless of the application location on any of the three machines.

In some implementations, these architectures permit the data management function to be placed in, and to function from, one machine only, regardless of the machine in which the application program resides.

IPAD priorities and funding limitations have dictated the sequencing of developmental activities; consequently, the greatest capability currently resides in the CYBER architecture, with some lesser concentration in the VAX. In 1981, much of this capability will also be implemented on the IBM 4341 machine.

IPAD documents have been created in the areas of (1) the user task environment and (2) a future "full IPAD" concept, as reflected in the preliminary design phase. In the case of the user task environment, documentation covers (1) reference design process, (2) interaction between design and manufacturing, (3) project management processes, and (4) data flow and data sizes consistent with the reference design process environment. Preliminary design documentation (reflecting future full IPAD) covers (1) "IPAD System Design Overview," (2) "User Interface Preliminary Design," (3) "IPAD Evaluations and Alternatives," (4) "IPEX Preliminary Design," (5) "IPIP Preliminary Design," (6) "IPAD Graphics," (7) "IPAD Geometry," (8) "User View of IPAD," and (9) "IPAD Level 2 Design." (Boeing documents D6-IPAD-70036-D, Volumes 1 through 9.)

Phase I, a subset of full IPAD design, has been developed. Figures 16 and 17 list the various software modules of this phase I development and indicate the engineering uses and capabilities of each.

ORGANIZATIONAL IMPLICATIONS

The emergence of large-scale integration of information and action, as discussed above, will affect not only the hardware and software architectures--the machines and the procedures for their use--but even more dramatically the organization and utilization of people as well. Present IPAD state-of-the-art developments have very broad implications for engineering and product-creation organizations. These organizations now face revolutionary questions with respect to the processes and resources of product creation in an integrated environment. These processes will have to be established, defined, and appropriately reoriented to capture the productivity improvement opportunities this integrated environment affords.

Each organization will need to review all the resources at its disposal. These include the people resources: engineers, programmers, and managers; the mechanical resources: computer hardware, both local and remote; the data resources: paper and electronic media; and the communications resources, both human and electronic.

The critical question that must be asked is: How shall we organize? Should we continue to be discipline- and action-

oriented (as is commonly the case)? Or should we reorganize to be task-, process-, and information-oriented? How can we best align these resources to control and direct them toward product creation at a higher level of productivity?

Figure 18 shows the most probable structure of an organization featuring integrated action and information. Most corporate product organizations are headed by a high-level executive charged with profit center responsibility for that particular product line. Given the state-of-the-art in computer hardware and software, it now appears very cost effective to combine the information resources with the people resources in an organization such as the one shown here.

CONCLUDING REMARKS

From the foregoing, it will be seen that productivity gains have resulted from the introduction of computer tools, albeit of limited information integration capability. However, if we are to meet the great productivity challenge of the 1980's, further progress toward new plateaus will depend on what can be achieved through large-scale information integration (LSII). This implies the further integration of hardware and software architectures, advanced IPAD-like system software, heterogeneous distributed processing with megabit communications, and application programs integrated within these architectures.

Organizational, engineering, and management philosophies and concepts are changing. Information will be a powerful new resource to be integrated with the usual people and machine resources. The implementation of these new concepts must be controlled and applied with utmost management skill if we are to achieve productivity improvements necessary to exploit the opportunities that will face us in the decade we have now entered.

REFERENCES

1. Miller, Ralph E., Jr.; McCaslin, John; Cost Effective Matching of Task and Computing Device. Journal of Structural Division--Proceedings of ASCE, January 1971.

2. Miller, Ralph E., Jr., Structures Technology and the Impact of Computers, 1975 ASM Winter Annual Meeting.

3. Hansen, S. D.; Southall, J. W., Feasibility Study of an Integrated Program for Aerospace-Vehicle Design (IPAD), Volume VII IPAD Benefits and Impact, September 1975.

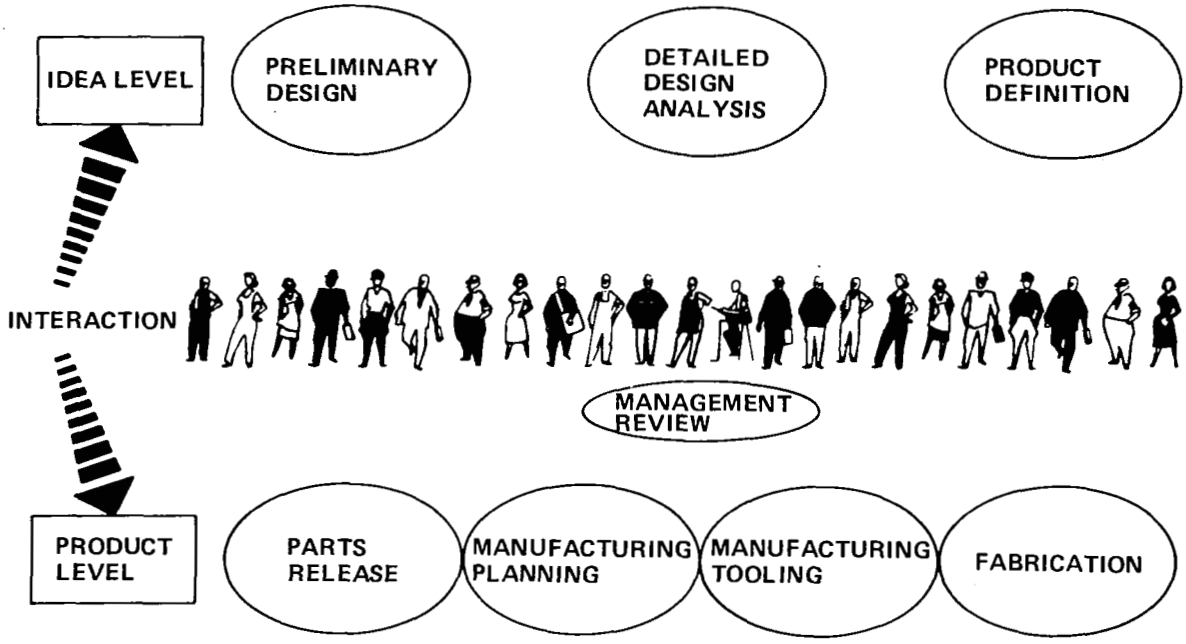


Figure 1.- Product information, quality and cost control.

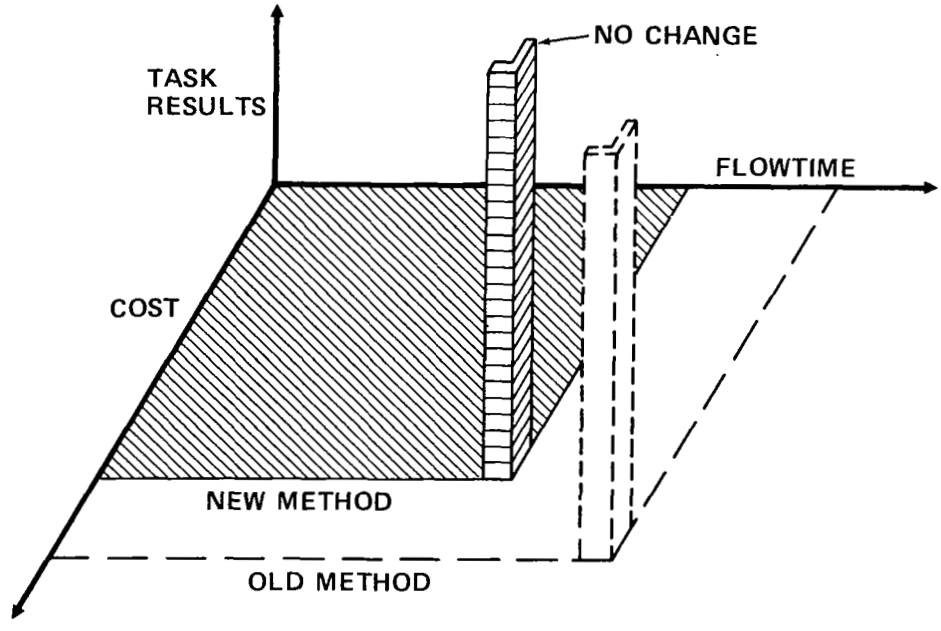


Figure 2.- Productivity improvement, reduced required resources.

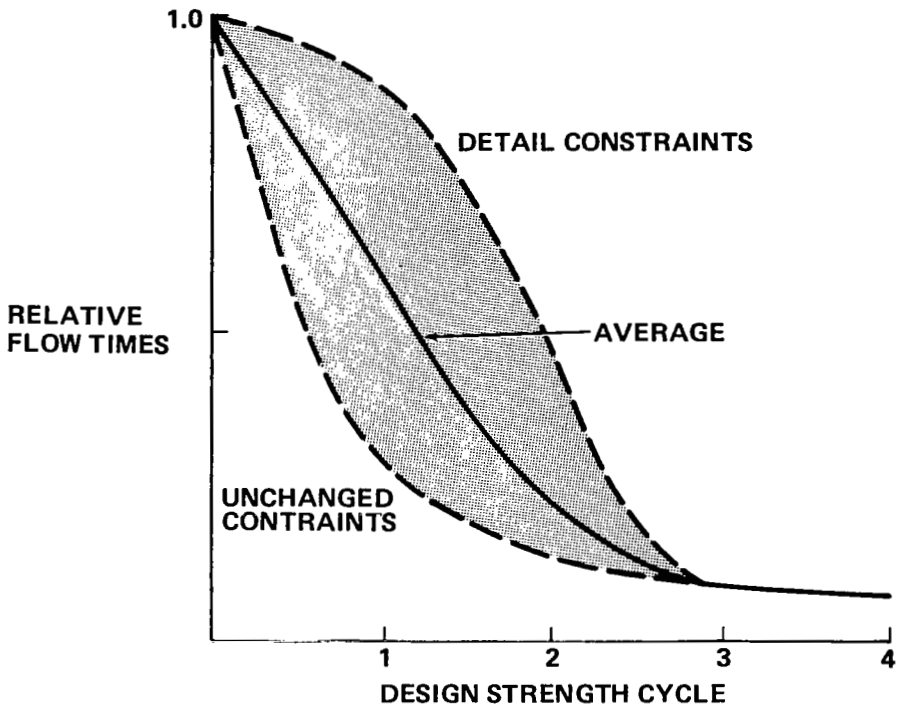


Figure 3.- Productivity improvement, reduced resources example.

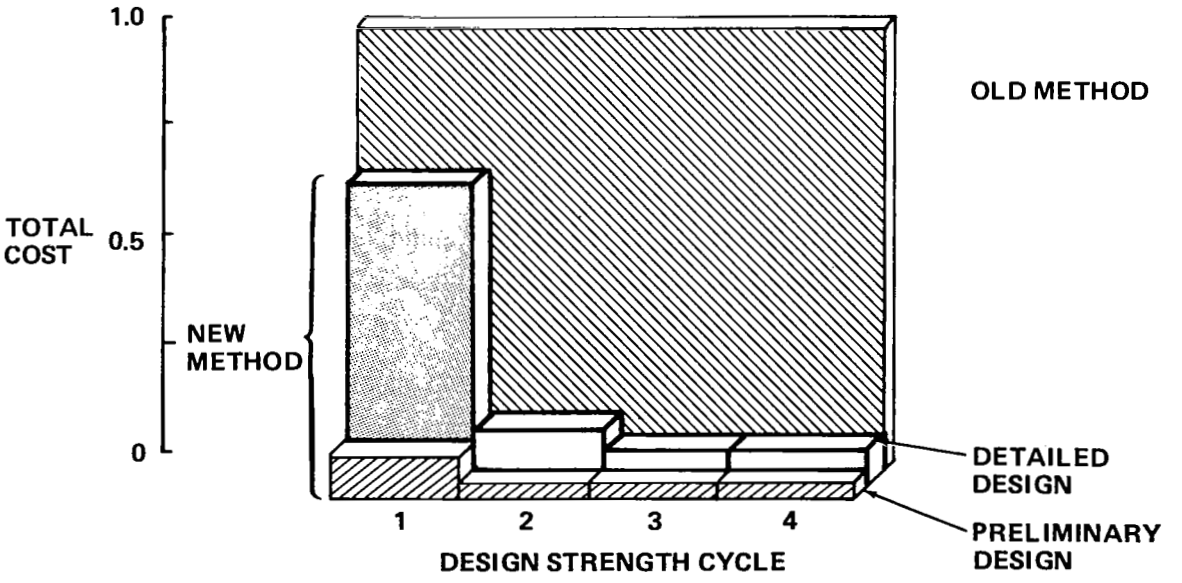


Figure 4.- Productivity improvement, reduced resources example.

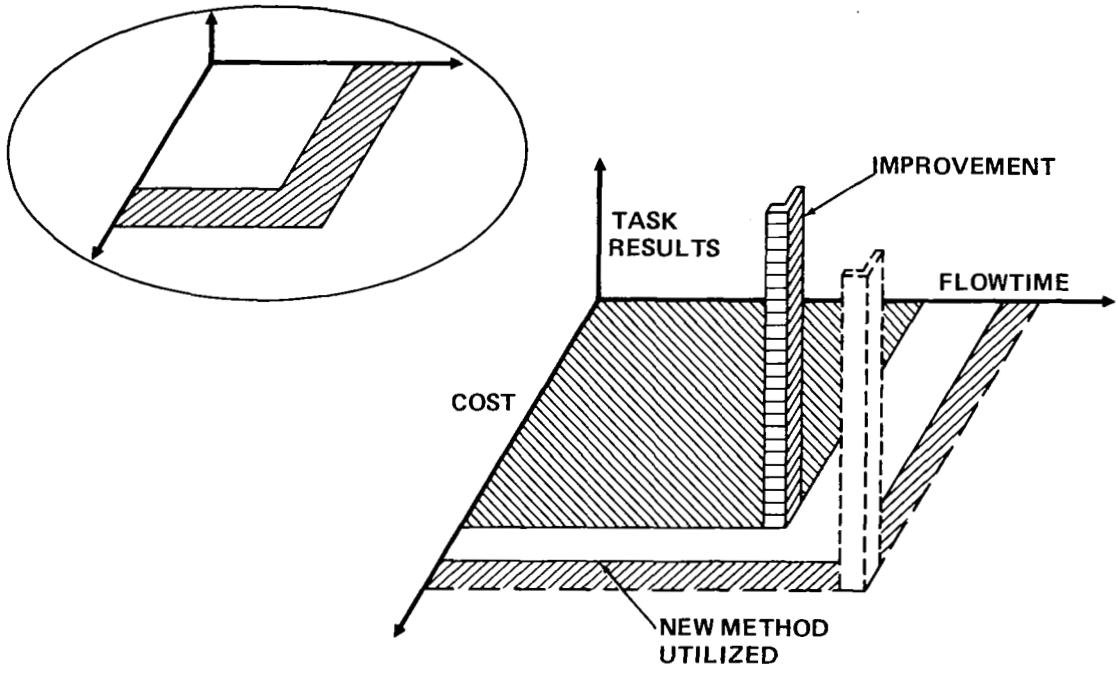


Figure 5.- Productivity improvement, improve task results.

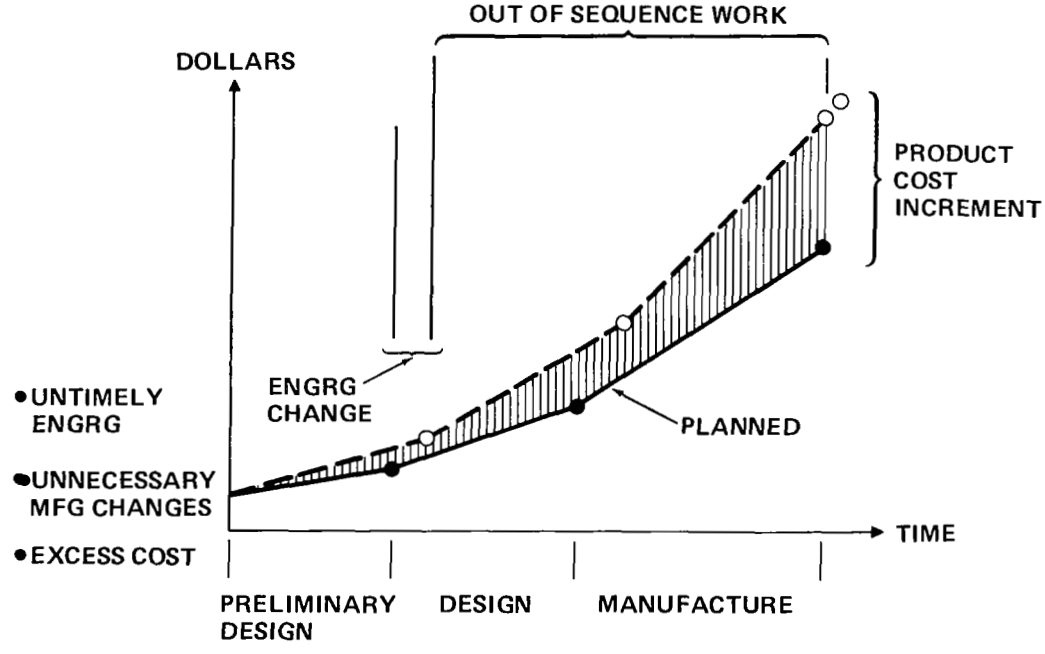


Figure 6.- Productivity improvement, reduce downstream costs.

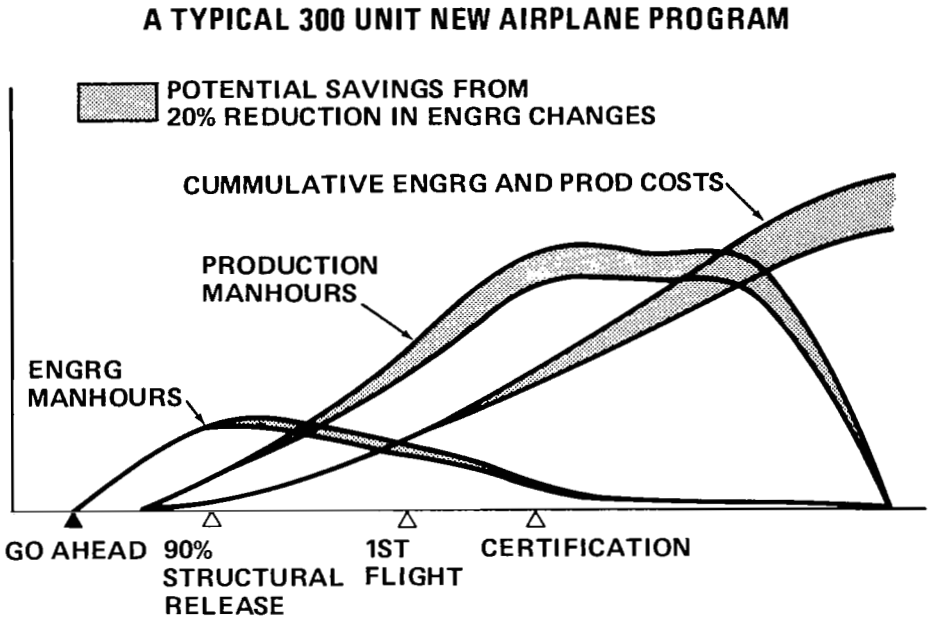


Figure 7.- Productivity improvement, reduced downstream costs potential.

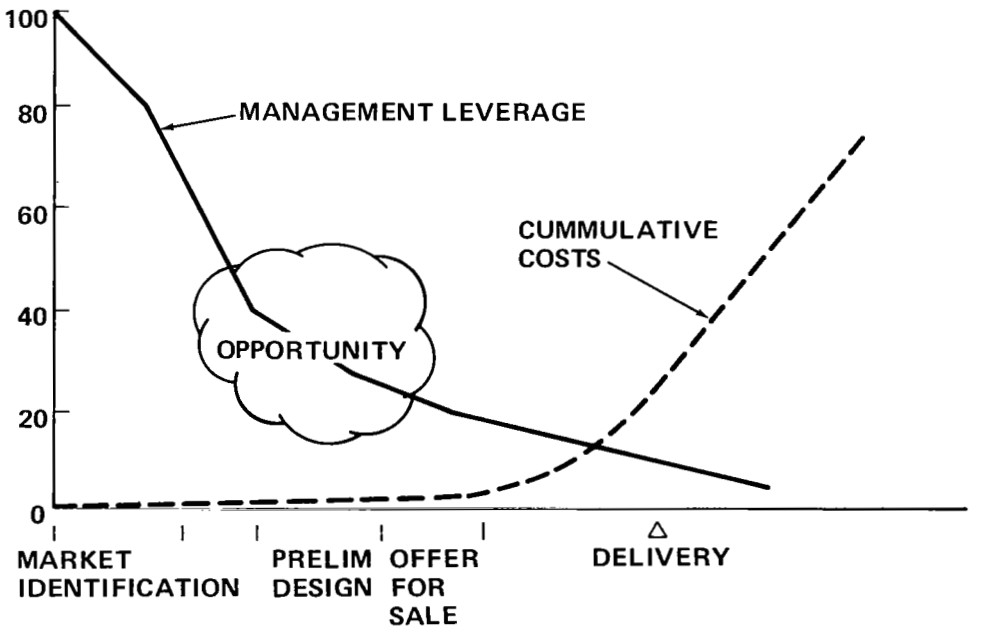


Figure 8.- Productivity improvement, reduce risk.

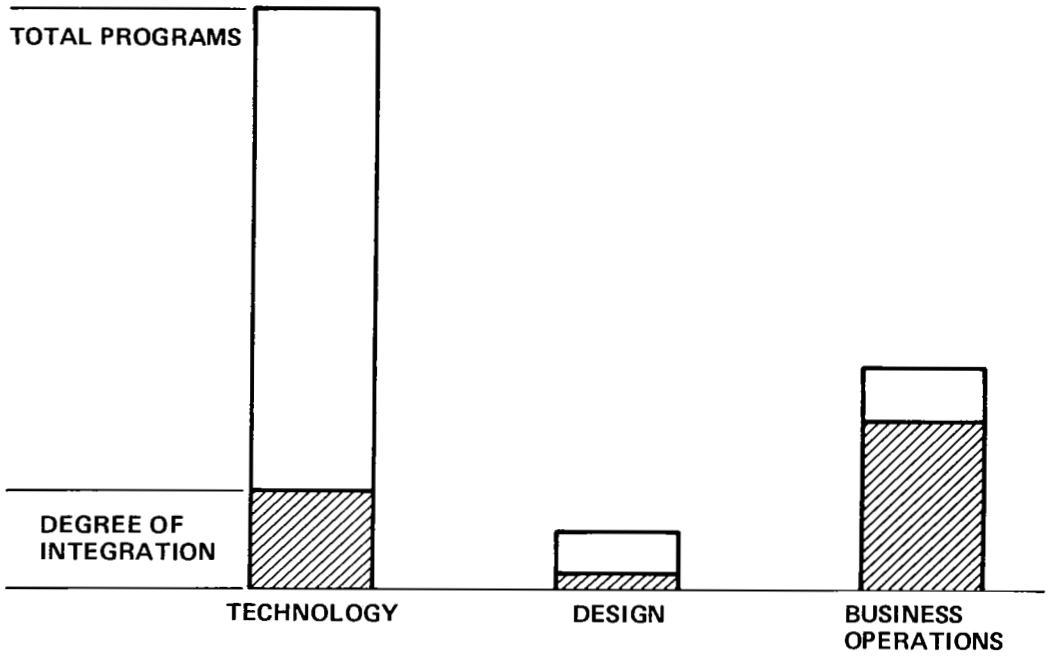


Figure 9.- Stagnating productivity, application program integration.

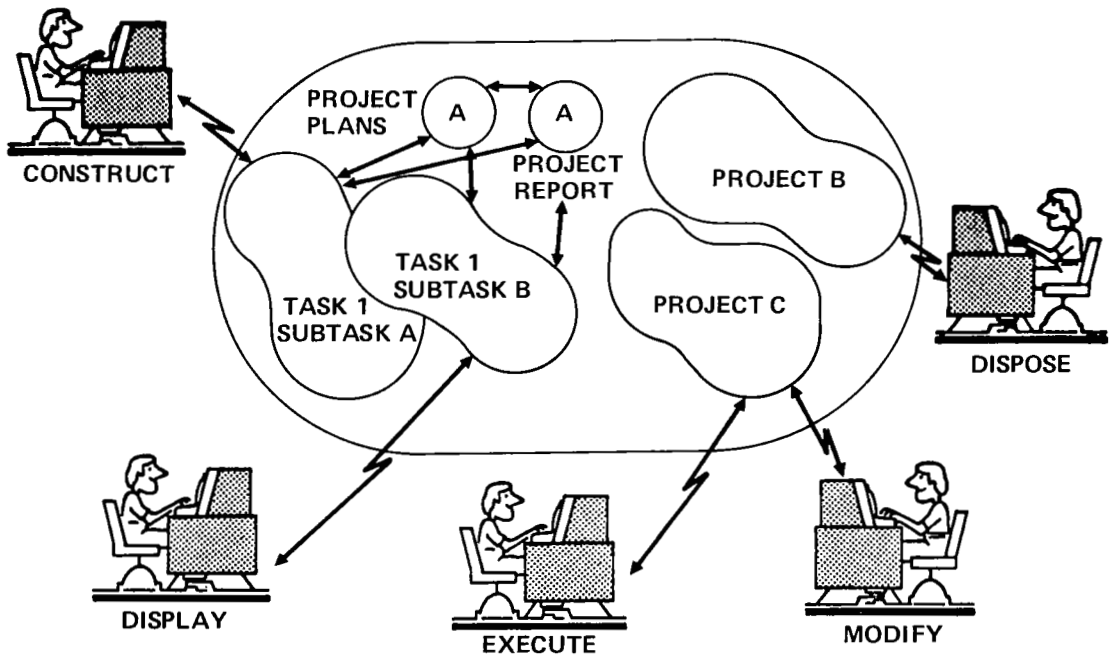


Figure 10.- Data base work relationships.

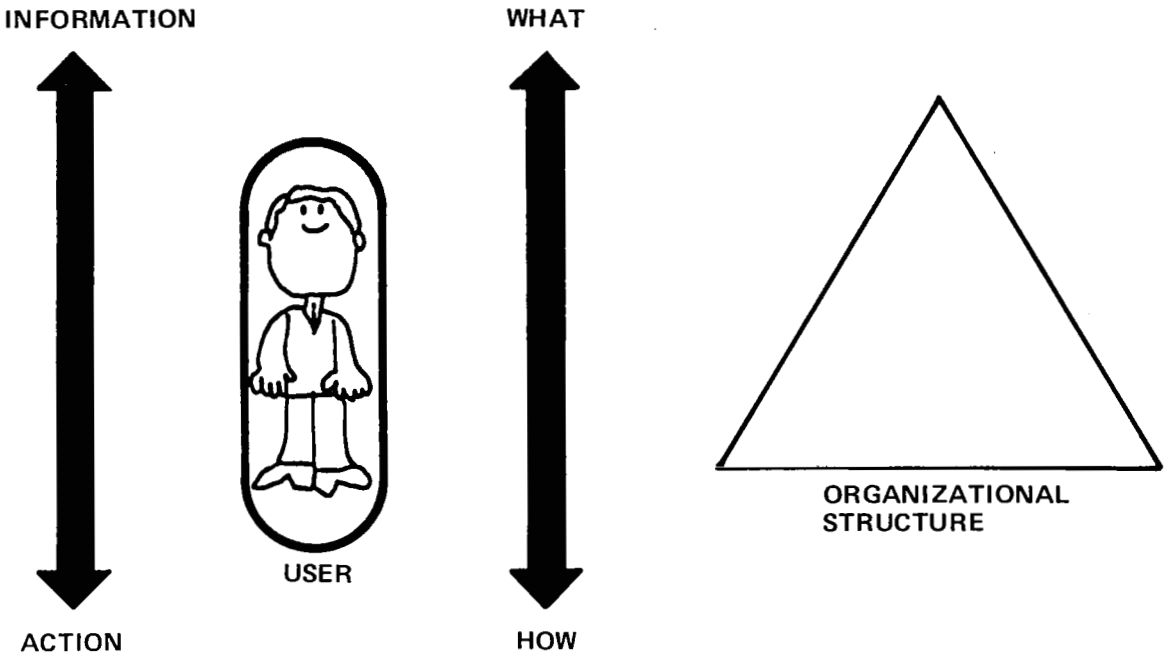


Figure 11.- Organizations - actions and information.

- LOTS OF PEOPLE/SPECIALIZED TASKS
- DATA BASES
- DATA SHARING (COMMUNICATION)
- STANDARDS (ESSENTIAL FOR COMMUNICATION)
- APPLICATION PROGRAMS

Figure 12.- Integration of action and information, system design issues.

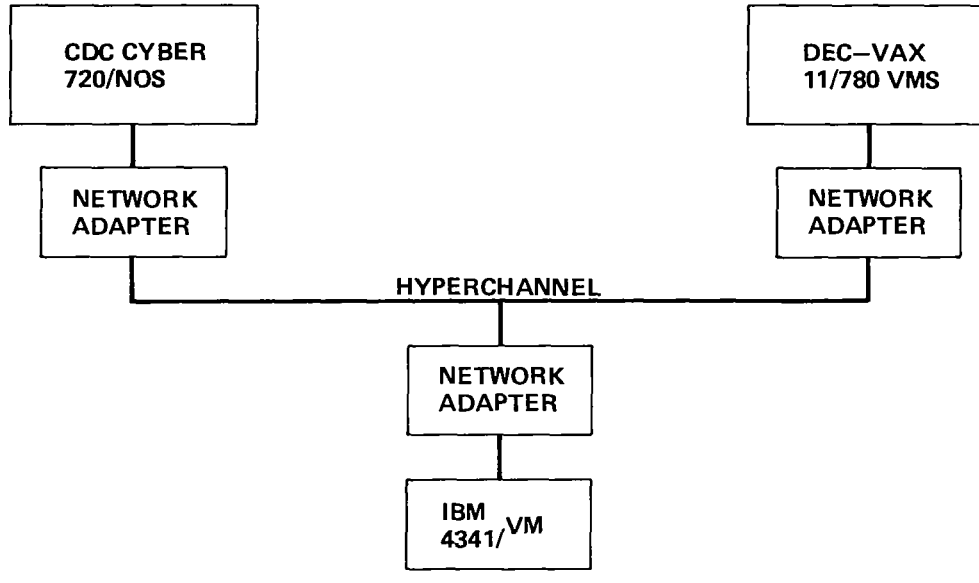


Figure 13.- IPAD products, phase I - hardware architecture.

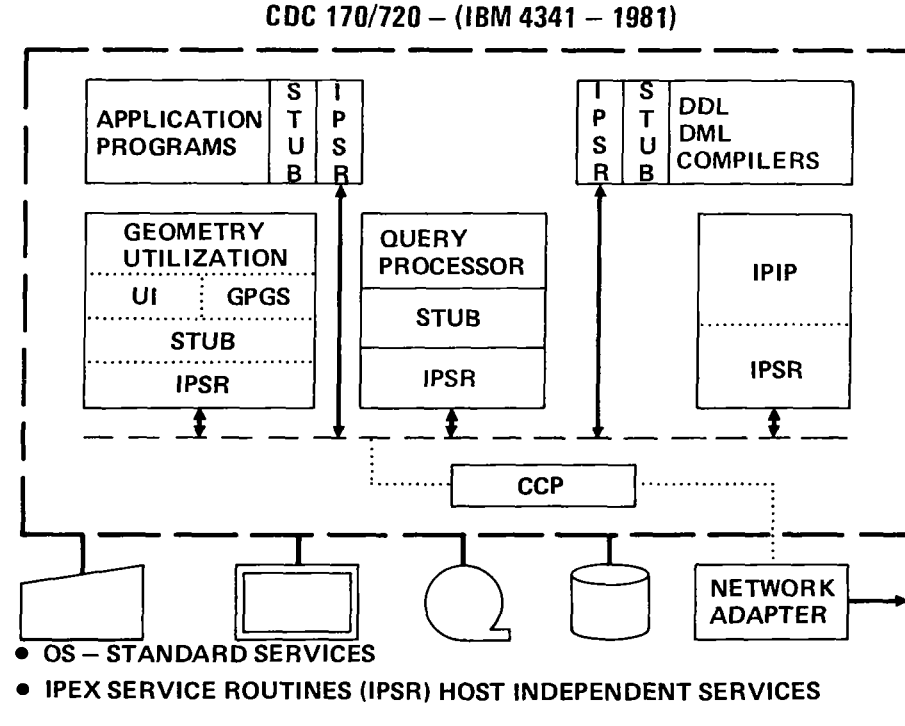


Figure 14.- IPAD software products - phase I.

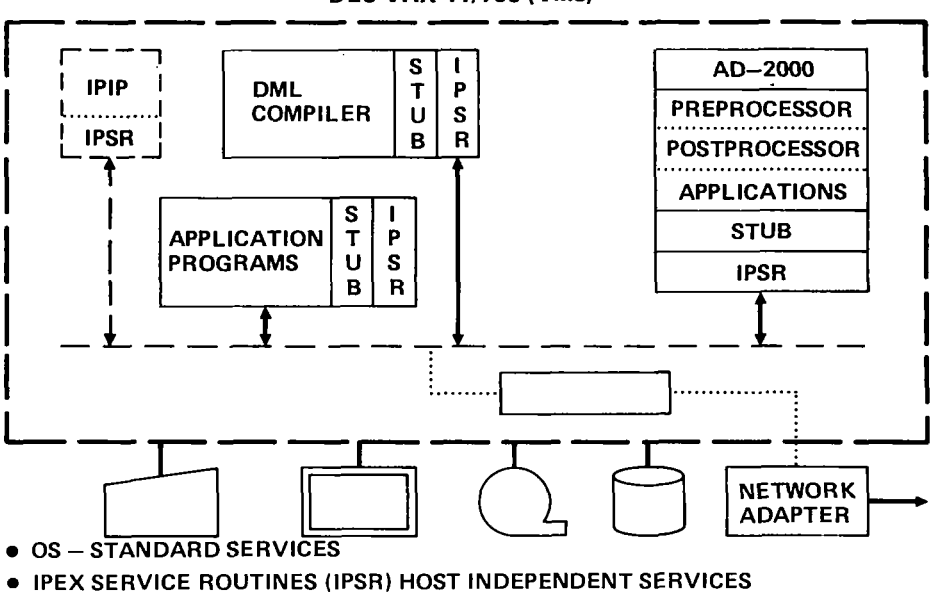


Figure 15.- IPAD software products - phase I.

PRODUCT	ENGINEERING USEFUL FUNCTION
NETWORK SOFTWARE	CYBER ⟷ DEC HIGH SPEED COMMUNICATIONS ● APPLICATION PROGRAMS ● IPAD FUNCTIONS ● HYPERCHANNEL HARDWARE
DATA MANAGEMENT SOFTWARE (MULTISCHEMA)	PHASE I – CYBER FUNCTION ● DATA DEFINITION & STRUCTURE ● APPLICATION PROGRAM USAGE ● USER QUERY ● DISTRIBUTED DATA BASE (USER TRANSPARENT) ● SPECIALIZED GEOMETRY SUPPORT DATA FROM DEC TO CYBER ● GEOMETRY CREATION ON DEC (DESIGN) ● APPLICATION PROGRAM ON CYBER (ANALYSIS)

Figure 16.- IPAD products - phase I.

PRODUCT	ENGINEERING USEFUL FUNCTION
EXPERIMENTAL GEOMETRY DATA BASE (RIM) SOFTWARE	CYBER FUNCTION CREATE & MANIPULATE (LIMITED) ● POINTS, LINES ● CURVES, SURFACES (PATCH)
GEOMETRY, GRAPHICS, USER INTERFACE SOFTWARE	DEC FUNCTIONS ● GEOMETRY & DESIGN CREATION • AD-2000 PRE/POSTPROCESSOR • APPLICATION PROGRAMS CYBER FUNCTIONS ● STANDARD GRAPHICS DISPLAY (GPGS) ● ANSI TRANSFORMATIONS (STDS.)
IPAD APPLICATIONS AND DEMONSTRATIONS	BROAD SCOPE ● ANALYSIS ● DESIGN AND MFG. INTERACTION ● QUANTITATIVE GOALS ● ENGINEERING PERFORMANCE DATA

Figure 17.- IPAD products - phase I continued.

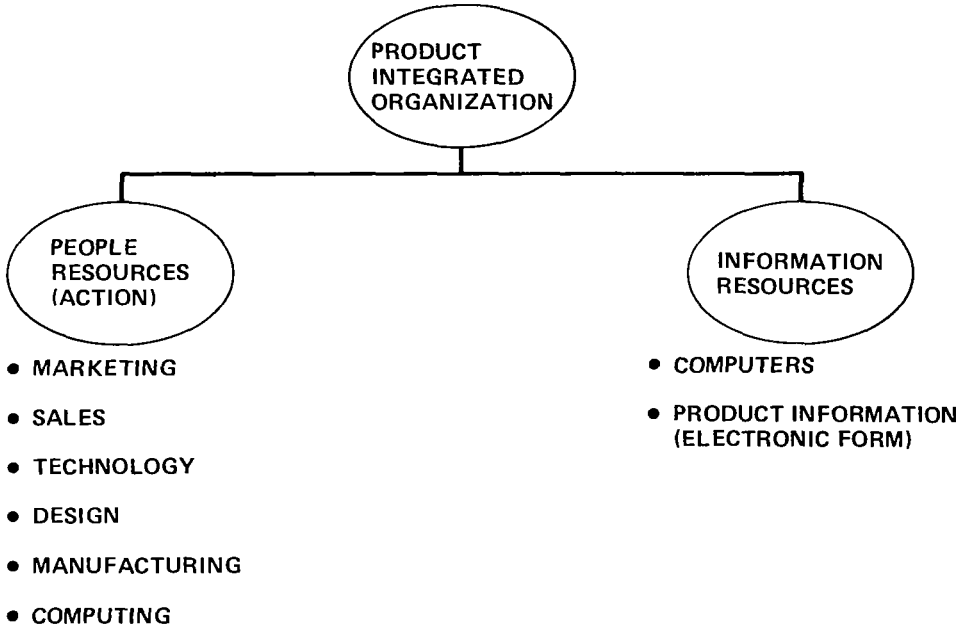


Figure 18.- Integration of action and information resources.

CASE STUDY - LOCKHEED-GEORGIA COMPANY INTEGRATED DESIGN PROCESS

C. Thurman Waldrop
R&D Staff Engineer
Lockheed-Georgia Company

SUMMARY

This paper presents a case study of the development of an Integrated Design Process at the Lockheed-Georgia Company. There has been a continuing growth of computer-aided design and analysis resulting in improvements in efficiency and capabilities but there are also problems such as suboptimization within each discipline, data handling, and data concurrency. The approach taken in preparing for the development of an integrated design process includes some of the IPAD approaches such as developing a Design Process Model, cataloging Technical Program Elements (TPE's), and examining data characteristics and interfaces between contiguous TPE's. The implementation plan is based on an incremental development of capabilities over a period of time with each step directed toward, and consistent with, the final architecture of a total integrated system. Because of time schedules and different computer hardware, this system will not be the same as the final IPAD release; however, many IPAD concepts will no doubt prove applicable as the best approach. Full advantage will be taken of the IPAD development experience. This case study, which includes forseeable problem areas, represents a scenario that could be typical for many companies, even outside the aerospace industry, in developing an integrated design process for an IPAD-type environment.

INTRODUCTION

The Lockheed-Georgia Company has been in operation since 1951 in a facility largely owned by the U. S. Air Force. The work force has ranged from 10,000 in 1958, up to 33,000 in 1969, and back down to approximately 10,000 at the present time. The Company is the prime contractor for the C-5, C-141 and C-130 military transports, JetStar business jet and numerous smaller projects. It has various modification projects and is a subcontractor for other Lockheed Companies and other aerospace manufacturers. Computers support all the basic operations, such as preliminary design, design analysis, production design, testing, tooling, fabrication, assembly, purchasing, product support, finance, and personnel. This paper is limited to the evolution of computer usage in the design process.

Lockheed-Georgia has steadily applied more sophisticated computer techniques to the analysis and design of aircraft systems as offered by the advances in computer technology. The engineering design and analyses

functions were initially conceived as a process of independent computations by each technical discipline. As design requirements have become more stringent, these independent computations have grown more detailed in scope, have required more input data, and have produced more output data. This growth in specialization and in the amounts of data generated have produced superior designs, but the management of this data has become costly and difficult. Over a period of time, the individual computer programs have been suboptimized for faster execution, and, in some cases, interface problems have been encountered. The means of handling data among the hundreds of individual computer programs in a design-analysis cycle is a major source of calendar time delays. As a result, Lockheed-Georgia has attempted to minimize the delays by optimizing the total engineering analysis cycle time rather than any one discipline methodology or computer program. Further, more extensive use was made of computer real-time access and data file management for communication of data, improved visibility, and ease of management across each discipline.

One example of progress in this effort is represented by the Integrated Structural Analysis System development. This operational system provides computer applications for more timely transfer and control of structural analysis data across the technical interfaces from aerodynamics through basic loads, dynamic response, flutter, and margin of safety analyses. This system substantially reduces the calendar time for a thorough structural analysis thereby reducing development costs. Another example is the ongoing project, Interactive Computer Program for Aircraft Preliminary Design Integration. However, these efforts represent only a small part of the job of integrating the total engineering design. Thus, a genuine need exists for a top-level integration of the overall design process to ensure timely and comprehensive control of the computerized design process at its many interfaces. The NASA IPAD program is a start in this important area, but the industry recipient must implement a comprehensive companion program to reconcile IPAD with his particular product line and approach to the design process. In preparation for taking full advantage of the IPAD Program products, Lockheed-Georgia initiated the Integrated Design Process Project.

INTEGRATED DESIGN PROCESS DEVELOPMENT

The first step was the development of a design process model for a thorough understanding of the interrelation and the sequencing of functions performed by the various disciplines involved in the design process. A multi-discipline team was established to assure that all disciplines were adequately considered and appropriately integrated into the overall process. Design, flight sciences, structures, avionics and information processing are represented by permanent members on the team; others are available on an "as required" basis. The design process model was developed graphically as a wall-sized display in a "Design Process Integration Control Room" as shown in Figure 1 so that it can be readily reviewed and discussed simultaneously by an appropriate group. This proved to be an excellent tool for showing the expert in a particular discipline how he

relates to the other disciplines and how his work fits into the overall design process.

This design process model was tailored after the model developed by the Boeing IPAD Team which is documented in IPAD Report D6-IPAD-70010-D. The IPAD model is divided into nine levels of activity as shown in Figure 2 for a commercial transport aircraft. The first stage, continuing research, is described as activity Level I. The long-term research stage continually provides new design procedures, technical analysis capabilities, and data for the design of aircraft. The second stage, preliminary design, is divided into four activity levels: design criteria selection (Level II), design sizing (Level III), design refinement (Level IV), and design verification (Level V). The third stage, product development, is divided into four activity levels: product detail design (Level VI), product manufacture (Level VII), product verification (Level VIII), and product support (Level IX).

The conversion of the IPAD model to apply to a military program process required a correlation of these nine levels to the program phasing required by the latest DOD System Acquisition Process reflected in DOD Directives 5000.1 and 5000.3 and OMB Circular A-109. The basic difference exists in that many of the nine levels of the IPAD process are overlapping activities wherein the DOD program phasing represents an end-to-end arrangement of activities with a decision milestone between each phase before the next phase can proceed. This is reflected in Figure 3 with an approximation of the relative correlation between the two types of programs.

Another major difference between the two types of programs is the timing of the first flight test of the aircraft. This point usually occurs in the military program during the Demonstration and Validation Phase (Block C of Figure 3) at which time the presently popular prototype flight test programs are conducted. This can be in the form of a competitive fly-off or a non-competitive proof of a previously selected aircraft before committing to a full-scale development and production program. The current high funding required for this approach is prohibitive for the civil sector to absorb, and, as a result, the first flight of future commercial aircraft usually will not occur until the first production aircraft. Other changes were required to the IPAD process in several areas to expand the process to show more of the significant functions and to reflect more complete interface of test requirements and results with the design functions. Additionally, extra routines were incorporated in the appropriate places for the consideration of load alleviation and flutter suppression features.

The IPAD design process reflected no wind tunnel test support for Level IV, design refinement. The only wind tunnel tests were shown as Level V, design verification, an after-the-fact effort. Therefore, Level IV was expanded by showing the earliest point in the design process when configuration development allows meaningful wind tunnel models to be designed and tested followed by the point when the wind tunnel test results are needed in the design process. Wind tunnel tests to support the design

process have always been a pacing factor, but reducing the cycle time through the integrated design process makes this pacing factor even more crucial. This highlighted the need to significantly reduce the time span from wind tunnel model design through test data reduction. As a result, a program was initiated for reducing this time span through computer graphic design, numerical control fabrication of the model, and improvements in data reduction techniques. This effort resulted in an ideal pilot program not only for integrating the design functions but also for demonstrating the feasibility of the automated transfer and use of data across the design/manufacturing interface.

The design process development was documented as a logic diagram with each function and decision point described by a narrative discussion. In addition, the logic diagram was computerized on the Applicon machine which is normally used for circuit-board logic design. This allows automatic printout and easily incorporated revisions. The design process was generalized to some degree as applicable to a generic subsonic military transport aircraft with a wing of moderate aspect ratio and with two to four engines located in conventional arrangements. With the use of the Applicon machine, the appropriate revisions can be quickly and easily incorporated to apply to a specific aircraft or to add special design features.

The (computer) application programs or Technical Program Elements (TPE's) required to perform each of the design functions in the design process have been identified. These TPE's are cataloged and classified as follows:

1. Operational - indicates that the program is in current use or has been used even though it may need some modification to apply to a particular model aircraft.

2. In development - indicates that the program is being developed but is not yet operational.

3. Not developed - indicates that a new program is desired but is not currently being developed.

The input and output requirements were recorded to aid in establishing data compatibility of contiguous TPE's to perform the functions. As a result, several activities were initiated to develop a run-stream of programs in various areas of the process. Also, a special effort was directed toward the interfacing of various geometry modeling programs such as GRADE, CADAM ®, Loft, 2-D and 3-D structural analysis models, and pressure distribution models in order that a more common geometry data base could be developed for all users. GRADE is the Lockheed-Georgia developed program, Graphics for Advanced Design, that enables the preliminary designer to perform his current drawing board task on a graphic display

® Registered Trademark, Lockheed Corporation

coupled with the full power of automation. As designs are generated, the program will automatically develop a mathematical model of the surface envelope. CADAM, Computer-graphics Augmented Design and Manufacturing, is a Lockheed-California developed program for design and drafting functions.

Another task was the definition of the features and requirements for those computing systems needed to handle all of Lockheed's engineering projects through 1985. The major task was to determine the existing machine dependency and geographical distribution of ongoing engineering computer aided operations and then define the needed changes in system features. A large part of the results of this task were incorporated into the update of Lockheed's computer capabilities.

The pattern of functional machine dependency and geographical distribution existing at Lockheed in 1979 is shown in Figure 4. The mainframe computers are all co-located in a central computing facility area, but are not directly coupled for any exchange of data. The work station for external remote processing is located about two miles away in an engineering facility area. The minicomputers are scattered throughout many different electronic and test laboratories. They are used for data acquisition and analysis in wind tunnels, full-scale static and fatigue tests, acoustic tests, and flight tests. Several minicomputers may be directly coupled in the same lab but none have networking features to other labs or to the central mainframes. Functionally, the engineering operations shown are machine dependent to a large degree. The various vendor computers have different word lengths and different internal character representations. Double precision is often required for accuracy on the 32 bit word IBM computer, sometimes on the 36 bit word UNIVAC, and rarely on the 60 bit word CDC computers.

The pattern of functional machine dependency and geographical distribution that Lockheed plans to have implemented in 1982 is shown in Figure 5. There are three major changes being implemented: (1) the mainframe computers and selected minicomputers will be interconnected for computer-computer data transfers, (2) the graphics scopes will be removed from the mainframe and remoted on minicomputers, and (3) array processors will be provided for rapid calculations. The major thrust of these changes is to provide for hardware integration and rapid data transfers between the various engineering disciplines and wide spread locations. The communications concentrator added to the mainframe computer was a major need defined in this study. These changes will also take more advantage of the newer, more powerful and lower-cost minicomputers. Several minicomputer vendors supply the required software for their machines to network and transfer data to other machines.

The long range goal of this system is to permit an orderly implementation of Lockheed's engineering computer aided operations and the IPAD software. The data base management system software could be implemented on one of the mainframe host computers and one of the minicomputers in each major lab installation. The data management software system will provide for the integration of design data as required. Thus, by 1982, major

improvements in design integration will have been achieved which will increase engineering productivity, reduce errors, add operational flexibility, and greatly reduce RDT&E costs in the design and testing processes.

Two specific areas within the design process have been identified as candidates for initial data base implementation. These areas are reliability/maintainability analysis and aeroelastic loads analysis.

The reliability/maintainability analysis area was chosen because of the extremely large volume of data involved and the necessity of automating the management of this data. Only one organization is involved in the preparation and use of this information. One important characteristic of this data is that it is fairly well structured and the data base can be implemented using the techniques associated with existing commercial data base systems.

The aeroelastic loads analysis area was selected because it is representative of the types of data and interfaces which a technical data base system should accommodate. The volume of data is large, the type and structure of the data is quite varied, and several different organizations interface with the data base.

Since no technical data base management system currently exists from which to gain expertise in defining technical data base characteristics, formal training in the area of logical data base design for commercial systems has been undertaken. The reliability/maintainability data base closely resembles a commercial data base and, therefore, its design can be initially modeled after a commercial system.

The risk involved in the application of IPAD is recognized. An IPAD which meets all expectations of performance, ease of installation, simplicity of interface between man and computer, and capability of supporting a totally integrated interactive design system, may founder in a particular installation if the proper environment and planning are not provided. This becomes particularly significant with the magnitude of the task to adapt a large existing design system to the organization and supervision of IPAD.

The Lockheed independent development of an integrated interactive design system must take full advantage of the benefit of IPAD, especially in the areas of data management and distributed computing with high-speed networks. Even though the distribution executive and the three-schema data base managers are desirable, the great generality afforded by the IPAD approach may result in performance degradation unacceptable in the Lockheed system. For these reasons, the progress of the IPAD Program is being closely monitored to assure that the future Lockheed efforts are commensurate with taking maximum advantage of any IPAD output. Plans are to participate in any IPAD user training, testing, and evaluation offered by the IPAD Program. Coordination will continue with computer equipment/software vendors to properly plan future needs.

The implementation of an integrated design process at Lockheed will be an incremental development of capabilities over a period of years. The development plan will be phased to progressively build the desired capabilities consistent with the availability of IPAD software. Since the data base management system is the heart of an integrated design process, the primary efforts will be in this area starting with the development of a data file management capability which will be designed to be compatible with an eventual evolution into a more sophisticated IPAD-type data base management system. An operational capability is expected by 1983. Further development of an integrated program and conversion of TPE's will provide an operational project capability of an integrated system by mid-1985.

SUMMARY OBSERVATIONS

Significant improvements in design productivity can be realized with the more efficient methods and operations offered by an integrated design process. This improved productivity can be manifested in a number of different ways. The most important is the reduction in design errors. Reductions in the design time span allows more iterations for design refinement within an allocated time period. The engineer is relieved of mundane tasks providing more time for creative work. Another significant advantage is that everyone uses the same current data base. Also, the appropriately planned and designed integrated design process provides the foundation for supporting computer-aided manufacturing operations.

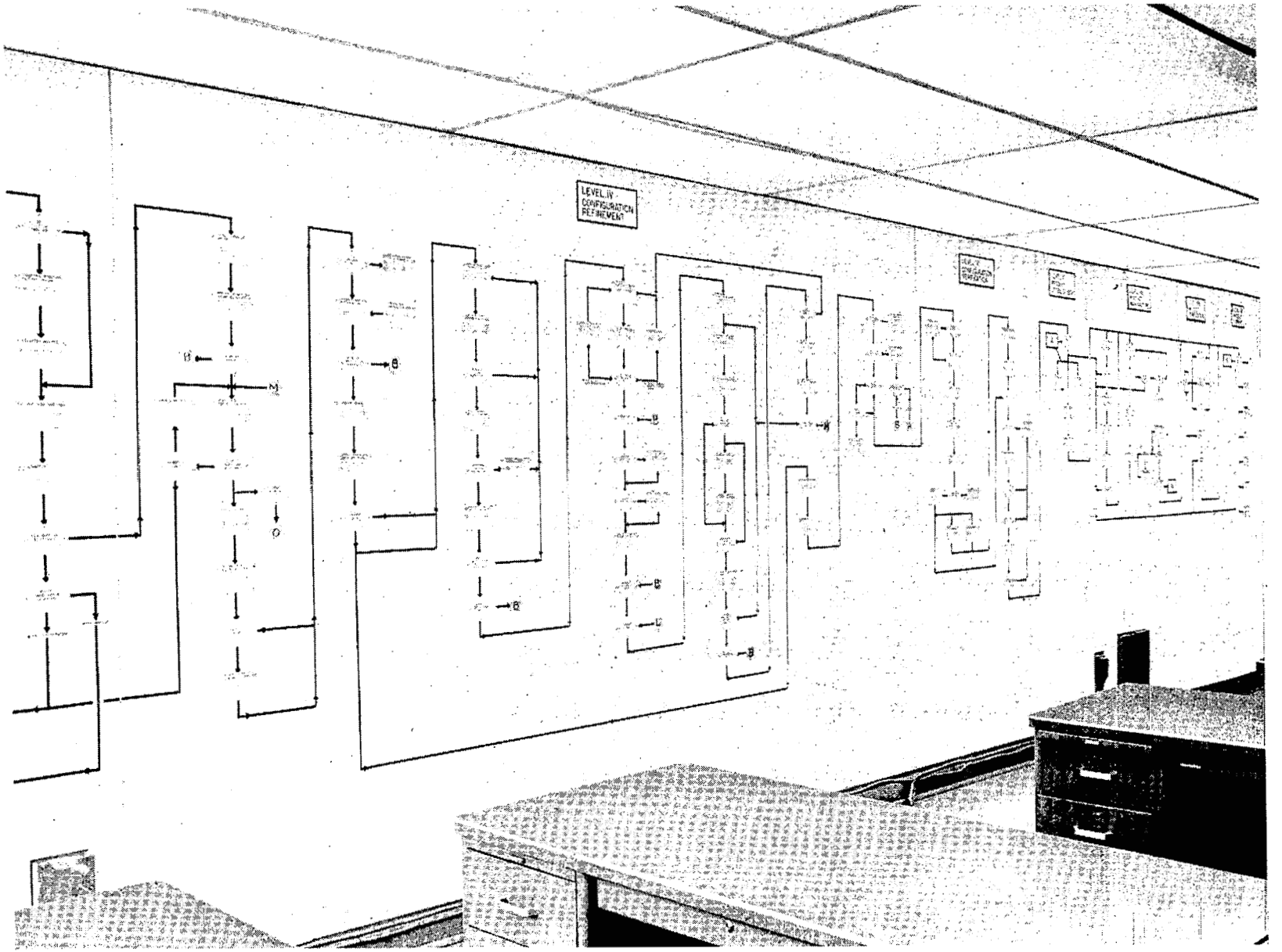


Figure 1.- Design Process Integration Control Room.

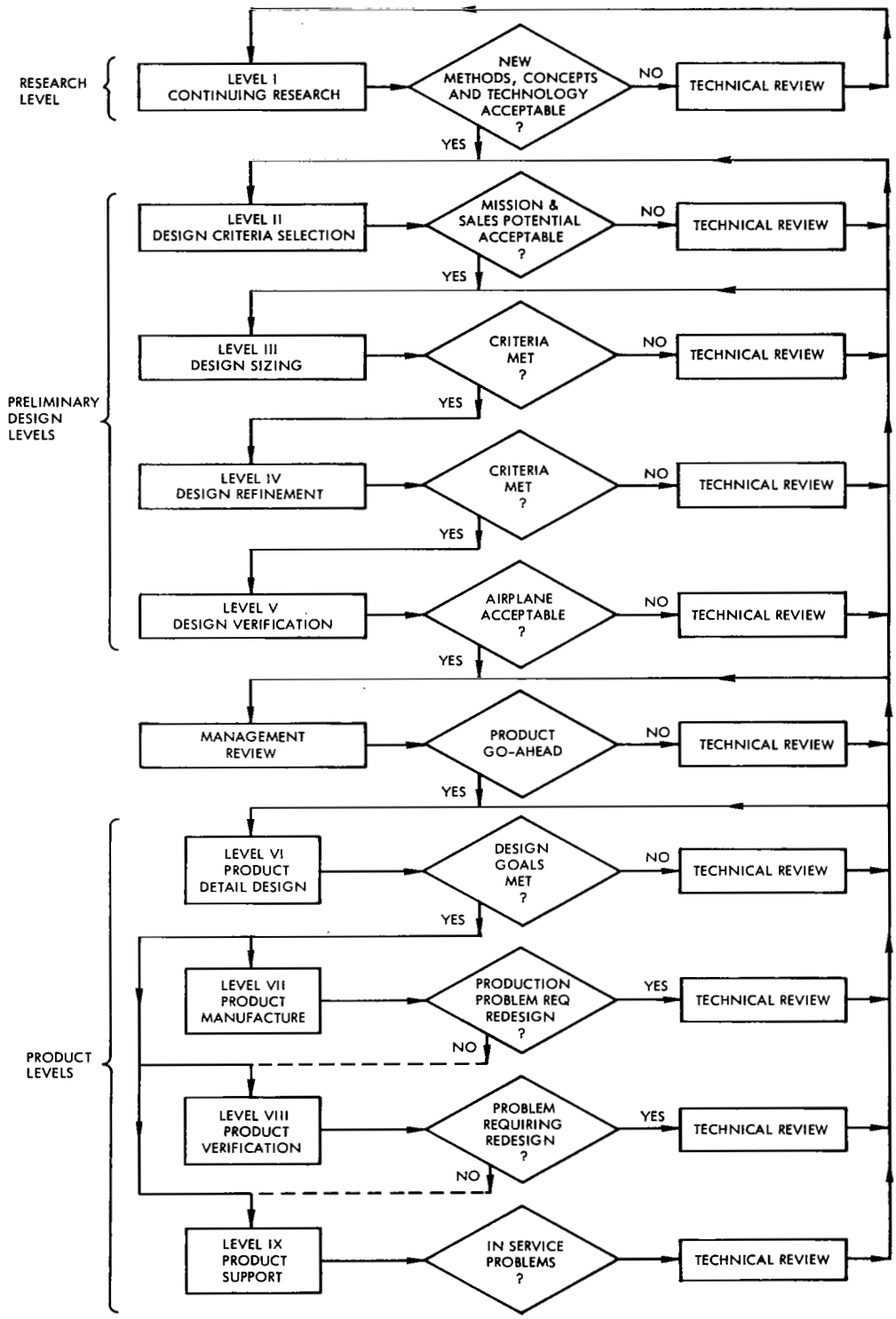


Figure 2.- Design process activity levels.

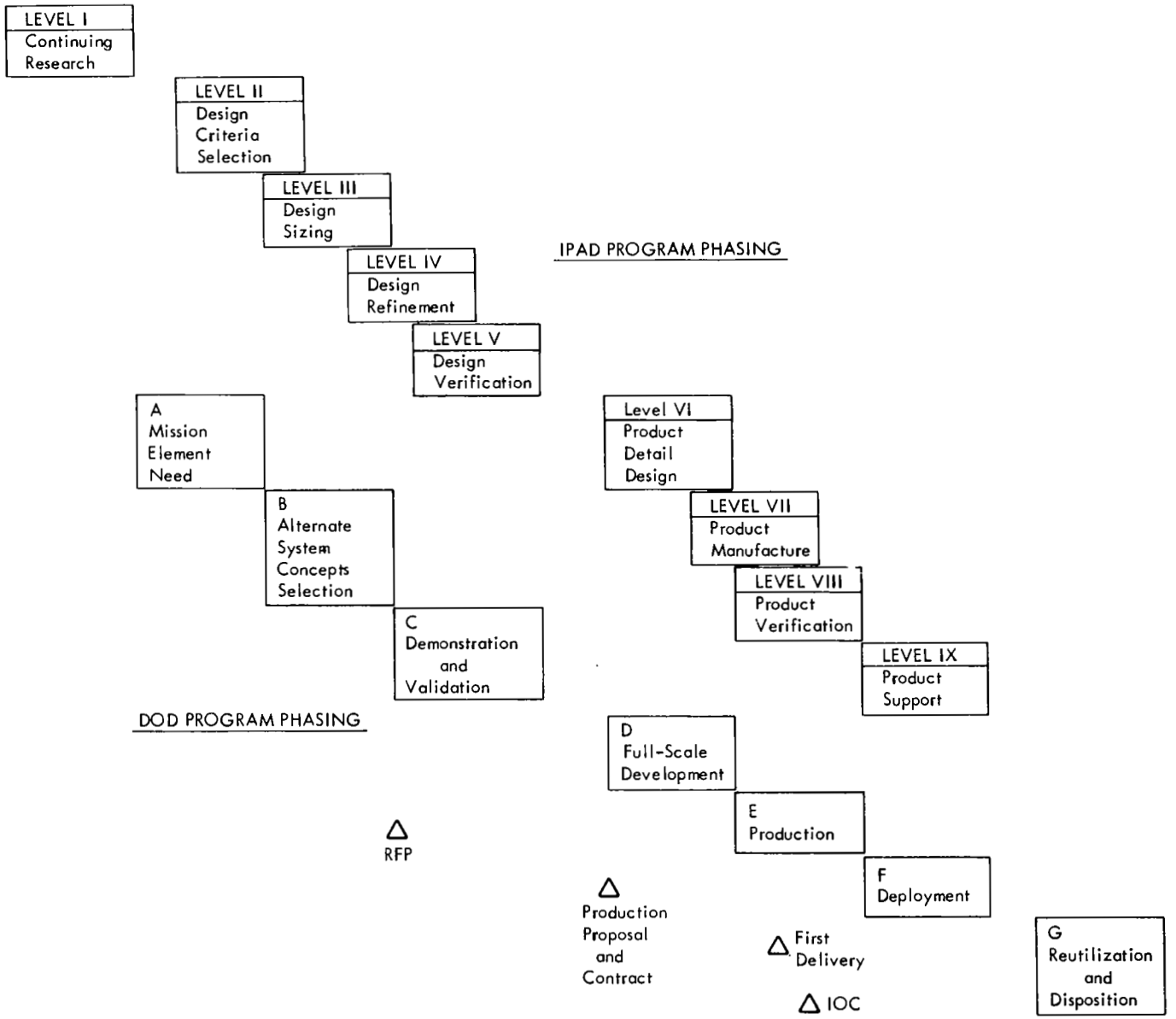


Figure 3.- Commercial vs military program phasing.

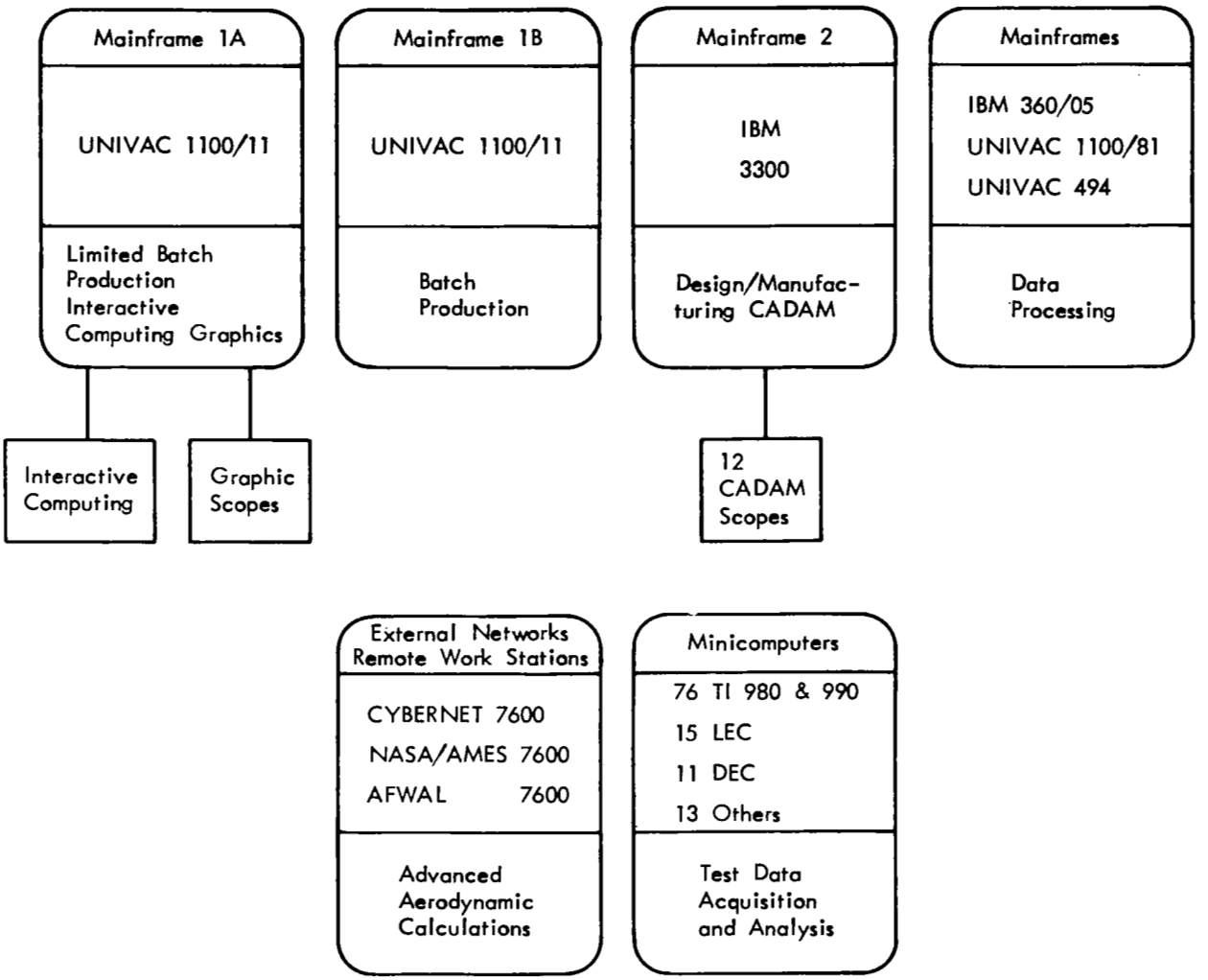


Figure 4.- Lockheed-Georgia Company computer systems in 1979.

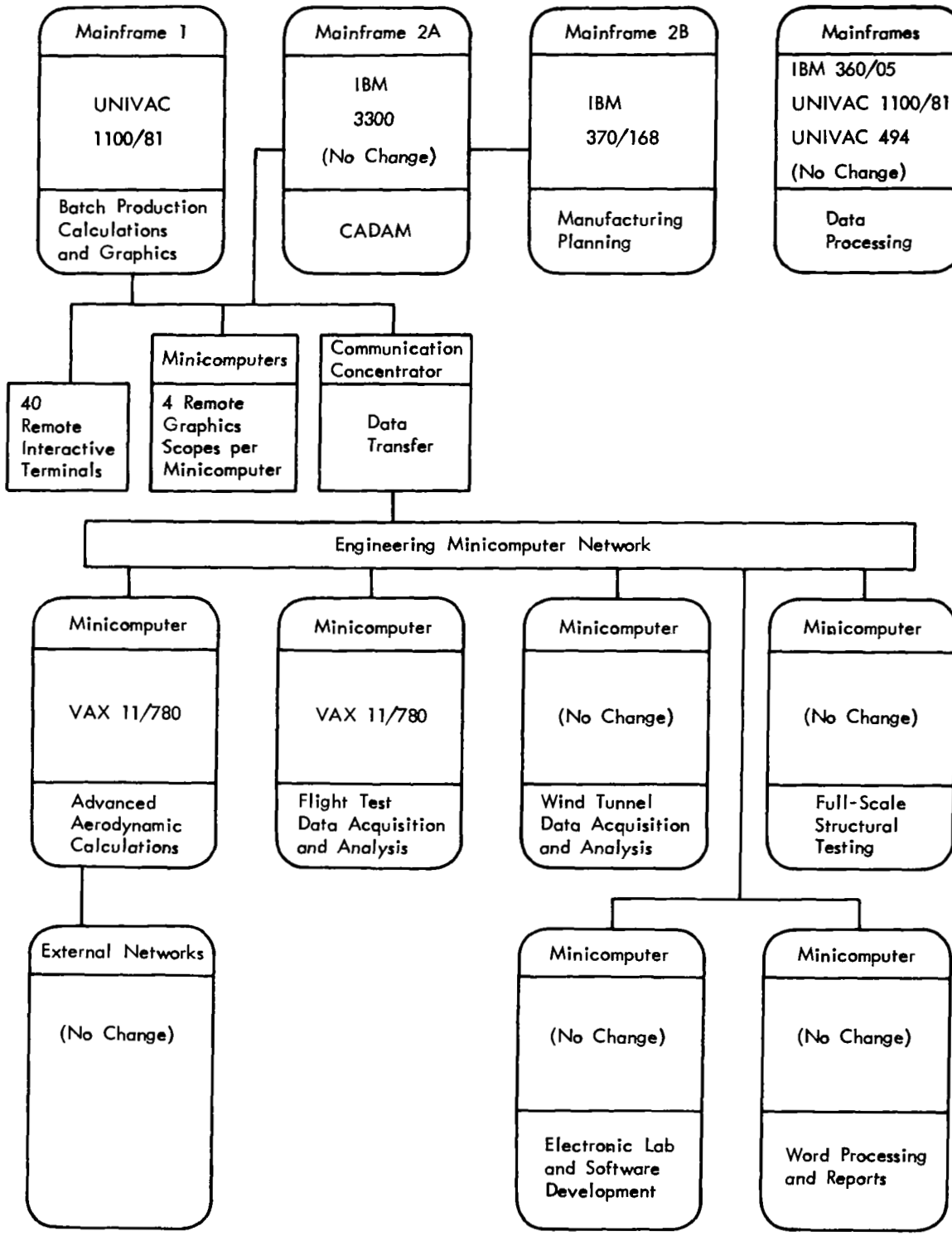


Figure 5.- Lockheed-Georgia Company computer system for 1982.

INTEGRATED CAD/CAM:

PROBLEMS, PROGNOSIS, AND ROLE OF IPAD

Dr. Edwin N. Nilson
Pratt & Whitney Aircraft Group

SUMMARY

Major technology problems impede the development and evolution of totally integrated interactive CAD/CAM systems. IPAD is playing an important role in the identification of these problems and is contributing significantly to their solution. It is the purpose of this presentation to examine some of these issues, look at the prognosis of obtaining effective solutions, and point up some of the past and expected contributions of IPAD to this technology.

INTRODUCTION

The acronym CAD/CAM for Computer-Aided Design/Computer-Aided Manufacture means many different things to different people. For some it covers the total application of computing to design and manufacturing and others it means something of much more recent vintage and in particular is associated somehow with interactive computing. Without attempting to resolve this debate, we use the term Integrated Interactive CAD/CAM to denote a special form of computing process in support of design and manufacturing in which:

(1) the description of a part and its properties are built up step-by-step in a common data base.

(2) this data is accessed and/or contributed to by all groups in Engineering and Manufacturing who are directly concerned with the design, manufacture, and quality assurance of the part.

(3) the process is continuous, beginning in Design and extending through Manufacturing and Quality Control.

(4) the fundamental mode of computing is interactive.

There is implicitly contained here the suggestion that such a system for the present deals only with technical computing in design and manufacturing as opposed to management and other supporting information systems. This connotation is intentional. Certain auxiliary information systems are usually required and are considered to be included; but it is strategically advantageous to establish first the level of integration indicated before the move is made to integrate the total computing effort--marketing, financial,

work scheduling, inventory control, product support, as well as engineering and manufacturing technical computing.

FUNCTIONAL DESCRIPTION OF I.I. CAD/CAM

In such a system, the integrated data base in a host computer or in a network of computers serves as the basis of communication. The schematic in Figure 1 represents the relation to users of such a shared data base, indicated by the long horizontal bar. Active portions of this data base are kept on-line and available for interaction by groups in both design and manufacturing.

By proceeding from preliminary design through mechanical design, design analysis and drafting, geometric descriptions of parts and their properties as well as those of subassemblies and assemblies of components are gradually developed in the common data base. Those elements of data required by Manufacturing from Engineering are available in the data base to manufacturing groups involved: process planning, tool design/make, numerical control (N/C) programming. These groups, however, not only interact with the design data in the data base, but build up their own data there for their own use and for communication among themselves. Automated inspection compares measured dimensions against design data and reports deviations therefrom.

It is in this way that the common data base becomes the fundamental medium for communication of technical information within and among design and manufacturing engineering groups as well as providing the working space for interactive computing.

The drafting process indicated will evolve over time. Initially, the drafting operation will continue to produce drawings with dimensioning and tolerances using information previously introduced into the data base, adhering to this traditional design-manufacturing communication for much of design. Ultimately this communication will be implemented essentially via the design data base. The role of drafting will change, possibly always completing dimensioning and adding tolerances, but now to input these into the computer instead of producing drawings per se. In the interim, drafting will probably continue to operate in both modes: inputting into the computer so that the part or object can be inspected by three-dimensional graphics, for example, and producing the conventional drawing as well.

The schematic in Figure 2 represents the interactive aspect of the operation. The user selects the appropriate deck or system to be used from the library of application programs (modules). Appropriate input will be called in from the data base, processed by the deck or program, the results displayed on a scope for the engineer's decision and subsequent action, and the final results stored back in the data base as required.

Other important elements appear here. In addition to the integrated data base and interactive interface with the user just described, integrated interactive CAD/CAM requires a computer executive structure by means of which

such a user interface, the data base interface, and indeed the integration of the entire structure can be effected. It is particularly important that the user interface be "friendly", that the computer command structure be convenient and easy to use, and that the response of the computing system be very fast.

There is one additional facility implied in the schematic. This is the library of interactive computing programs and modules available. These consist not only of specific applications programs employed in the design/manufacturing process but general facilities as well. Thus it might include finite element systems with preprocessors and post-processors for design, and N/C graphics programming modules and machine post-processors for manufacturing. It will include, as well, geometric modeling and graphics capabilities, systems to permit interactive drafting, for example, or sketching by the designer, or the modeling of a machining process by a process planner. It will include conventional N/C programming tools and post-processors for these.

TECHNOLOGY SUPPORT FOR INTEGRATED INTERACTIVE CAD/CAM

There is a great deal of perplexity as to what is required to build an integrated interactive CAD/CAM (I.I. CAD/CAM) system and how to go about this. This issue is further complicated by the rapid proliferation of turnkey interactive graphics drafting systems like ComputerVision, Applicon, Gerber, CALMA, etc., many of which are purchased under the impression that they constitute, or form the basis for, an integrated CAD/CAM system.

These mini-computer-based IGDS's (Interactive Graphics Drafting Systems) are moving into a vacuum created by the lack of I.I. CAD/CAM structural plans and technical requirements in addition to the dearth of suitable computer operation systems, data base management systems (DBMS's), geometric modeling systems.

During the late 1960's and early 1970's, guidance and support by the large computer vendors for integrated interactive CAD/CAM have been conspicuous in their absence, and interested customers for the most part have not known which way to turn. It was to provide the missing technology and structure that NASA introduced IPAD in the middle 1970's and the NASA/Boeing IPAD project was initiated in early 1976.

We would like to examine some of the specifics of this picture, to indicate what we see as the major technology problems remaining, to comment upon the prospects for their solution, and in this framework to point up what we see as IPAD contributions to date and to be expected in the near future. We shall consider three problem areas which admittedly may be neither mutually exclusive nor complete: computer technology problems, design and manufacturing engineering technology problems, design/manufacturing technical communication problems.

COMPUTER TECHNOLOGY

We consider here in particular computer operating systems, graphics, data base management systems, distributed processing, integration of IGDS's.

At the outset we point to a major IPAD contribution. Note that an early IPAD objective was a kind of "super operating system" for CDC and IBM computers which would permit the construction of an integrated interactive CAD/CAM system by the user. In this connection, the Boeing IPAD team systematically and thoroughly assembled a set of basic requirements--an essential step but one which had never previously been done.

The only computer operating system which came close to meeting requirements for I.I. CAD/CAM was an IBM system, VM/CMS, which had relatively little support within IBM itself, and found its principal proponents, other than one or two U.S. aerospace companies, located in Western Europe. This system is excellent with respect to simplicity of use and "friendliness" for the engineer-user, time sharing, virtual memory providing each user with the equivalent of a large computer, communication between users, and a data base capability certainly adequate for starters. While other systems had many, if not all, of these capabilities nominally, the "feel" of these systems was not right for I.I. CAD/CAM.

Now the outlook is decidedly more promising. Spurred on in part by IPAD's findings, Control Data, Digital Equipment, UNIVAC, and others are improving their time-sharing operating systems to the point where they will probably be competitive. IBM, moreover, has decided not to abandon VM/CMS and is giving it a fundamental role in its approach to distributed processing.

Graphics technology is moving ahead rapidly and appears not to impede the development of integrated CAD/CAM, at least in respect to hardware. The increasing use of microcomputers to provide "intelligent" scope terminals is beginning to reduce the burden upon the host computer. Color graphics are becoming more common.

Three-dimensional graphics subsystems, supported by mini-computers, have been available for some time with which continuous translation, rotation, and zooming in 3D space can be smoothly carried out. These have applications in providing a 3D viewing of a design and in permitting inspection of NC (Numerical Control) cutter paths in 3D, as well as in many other areas.

Graphics software for the user lacks standardization, but this fact is more a nuisance than a technological block to I.I. CAD/CAM. There is a group, under the auspices of the National Bureau of Standards and the Air Force ICAM program, working on this problem: IGES (Initial Graphics Exchange Specification). There is a long way still to go, however.

(There is a potential health problem developing associated with the use of interactive graphics and cathode-ray displays generally, the long-term impact upon I.I. CAD/CAM of which remains to be evaluated. Some people who are

exposed for long periods of time to this type of radiation seem to be developing cataracts.)

The need for an effective data base management system (DBMS) for scientific/engineering computing is a major technology obstacle. Existing systems have been designed to handle data bases for information systems rather than design/manufacturing engineering technical systems. Here the data, especially geometric data, is heavily hierarchical in nature, a characteristic not effectively accommodated by earlier systems. We have set forth elsewhere (Reference 1) both design and manufacturing requirements of a DBMS for engineering technical data as they are perceived at this time. Both ANSI (American National Standards Institute) and CODASYL (Committee on Data Systems Languages) have come to grips with the problems of establishing specifications for such a DBMS.

Here, with the DBMS, IPAD is currently making a major thrust, concentrating much of its effort upon a relational-type data structure, not too dissimilar, perhaps, from the IBM's System R which is still in the development stage.

ITAB, the Industry Technical Advisory Board of IPAD, took the initiative in 1979 of establishing with the Air Force a joint IPAD/ICAM workshop to consider DBMS requirements, problems, solution. This workshop was held in Dallas on April 24, 25, 1979, in a series providing continued communication on this critical subject. (See ref. 2.)

IPAD is also currently concerned with distributed processing and high-speed networking. Distributed processing will impose additional requirements upon both computer operating systems and DBMS's. Local data bases may well be established at computing nodes in a network, and the user--engineer or programmer--must not have to worry about the location of particular blocks of data. In a related requirement, a user with a terminal into one computing node should be able to use a computing system in another node and be free as to the disposition of results.

Distributed processing technology presents no serious obstacle to the implementation of I.I. CAD/CAM at its present stage of development, and seems to be moving steadily ahead with computer vendors. Total integration of equipment of several vendors, however, will present problems.

There is a situation which appears to be developing into a major obstacle to integrated interactive CAD/CAM. This arises out of the proliferation of mini-computer-based interactive systems, including interactive graphics drafting systems (IGDS's). IGDS's designed and marketed by ComputerVision, Applicon, Gerber, CALMA, and others are intended to capture the lead time and labor reduction advantages at the design/manufacturing interface. They offer an interactive drafting capability with a certain NC programming capability; in addition, there is usually some structural analysis possible.

These systems are frequently advertised as providing the basis for an integrated interactive CAD/CAM system, and many companies are introducing them

for this reason. Networks generally exist for connecting several of these together, and the mini-computers seem gradually to be growing into maxi-computers. The CAD/CAM technology which they offer is developing fairly rapidly.

They have some serious disadvantages. As turnkey systems, they are difficult to adapt to, or conform to, an existing design/manufacturing engineering philosophy. They are generally inflexible, and the vendor keeps tight hold upon the software. It is not feasible to try to build into such systems additional aspects of the design function which are essential to the design process. Further, they will not fit simply into the distributed processing systems which the large computer vendors are developing for themselves although they appear to be on the threshold of moving into their own distributing processing mode.

Companies with large numbers of IGDS's and other systems currently face the problem of integrating these systems. Integration of computing facilities, data bases, and other functions is a very formidable task. On the other hand, the IGDS is in many situations the answer to interactive CAD/CAM for the small manufacturer.

DESIGN/MANUFACTURING ENGINEERING TECHNOLOGY

Here there are several problem areas, for the most part related to the function and needs of the process planner. Computer-aided process planning is in its infancy with major obstacles ahead. Many of these are associated with group technology, or parts classification, and the latter, indeed, in itself, represents an engineering technology problem.

Advanced NC programming techniques, especially NC graphics, are already in extensive use and have a dramatic effect on reducing lead-time-to-production in NC. These also have an important relation to computer-aided process planning.

At the core of most, if not all, of these problem areas is the technology of geometric modeling which is not only in need of standardization at its current level of development but will require a major breakthrough before truly natural and effective tools are available to the process planner. We consider these probelms in reverse order.

There are two major problems in geometric modeling, one short-term and one long-term.

The short-term problem is the need for good geometric modeling/interactive drafting systems for large-scale computers. With a few exceptions, like the Lockheed's CADAM system currently marketed by IBM and the McDonnell Douglas CADD system as distributed by McDonnell Automation, there is a dearth of such systems for large computers. There are a few encouraging signs. Control Data Corp., Digital Equipment Corp., and UNIVAC are making significant efforts to

to adapt Hanratti's AD2000 system to their computers. IPAD is following a similar path. Improvements in CADAM may be forthcoming from Lockheed, and there are some signals that IBM may be working on such a system.

The availability of such systems would open up new alternatives to users who have been more or less restricted heretofore to mini-computer-based systems.

The major long-term problem is solid geometry modeling. The central figure is the process planner who requires a clear and unambiguous communication from engineering design, and who needs an effective means of describing the changing configuration of a block of metal as the machining process progresses. The conventional orthographic and isometric projections fall seriously short of the mark.

We begin to see something of the nature of the change which will appear in what Voelker of University of Rochester has developed with PADL (Part and Assembly Description Language) or Fuchs' work at Technische Hochschule Aachen. Here a solid is described and a machining process spelled out by the addition and subtraction of a relatively small number of "primitive" solid shapes: rectangular parallelepipeds, wedges, circular cylinders, cones, etc., which are added and/or subtracted to form the desired part shape. Drilling a hole, for example, is equivalent to subtracting a circular cylinder.

Other organizations in East Germany, Hungary, and Japan have developed similar systems. Two conclusions can immediately be drawn: (a) PADL-type 3D geometric modeling is not the final answer--a continuous deformation of a PADL shape is usually outside the system; (b) even where PADL-type languages are effective, either the designer must be coerced to change his way of life, or there must be provided sophisticated translation procedures to bridge the gap between designer and process planner. (See ref. 3.)

A wholly different approach is that of the American National Standards Institute Committee Y.14.26 wherein an arc is swept out by the continuous movement of a point, a surface by the movement of an arc, and a solid by the movement of a surface. Any effective approach to solid geometric modeling will transcend both of these approaches.

Interactive Process Planning is dependent for its development upon all of the other technologies: group technology, NC graphics, geometric modeling. A parts classification system which groups parts with similar process plans is essential to the retrieval, and subsequent modification or refinement of such a plan. For the special case of parts with rotational symmetry, interactive process planning is very much simplified (the cross-section profile is 2D): thus, both G.E. and P&WA have systems which are about ready to go in this area--G.E. has its Rotating Parts Operation (RPO), P&WA has its Computer-Monitored Process Planning (CMPP).

NC Graphics, viewed by many companies as the route to shorter lead-time-to-production and higher NC machine tool utilization through reduced number of tape tryouts, for example, provide the process planner with an interactive

means to choosing cutter paths and other machining characteristics. Although subject to the limitations imposed by conventional geometric modeling, NC Graphics will go far toward advancing the state of the art in process planning. In 2D (i.e., 2-axis machining or turned parts), the geometry represents no block.

In those areas of machining in which a PADL-type language is applicable, significant savings in lead-time-to-production are realized. There are obvious problems, however, in having two such radically different geometric modeling systems existing side by side in a manufacturing business.

Group technology has its application to design, where it has important uses in the classification and retrieval of drawings or design processes. In manufacturing, its applications range from the classification and retrieval of machining/fabrication processes which are part-oriented to the classification and retrieval of tools and fixtures in fabrication and assembly which may require classification on the basis of function as well as part.

Fundamental conflicts arise between requirements of design and those of manufacturing. Design's classification based upon geometric configuration and Manufacturing Engineering's use based upon machining/fabrication process do not coincide. For turned parts--parts with rotational symmetry--these conflicts are a minimum.

For aircraft engines, there is a natural parts classification based upon location of the part in the engine: we employ an "Engine Area Code" specifying the component, subcomponent, in which a part is located, down to the individual part.

DESIGN/MANUFACTURING TECHNICAL COMMUNICATION

In integrated interactive CAD/CAM technology, the shared data base provides the means of communication within and between groups in design and manufacturing as well as between design and manufacturing. The description of a part is developed in design in the data base, and those elements of data to be transmitted to manufacturing are "signed off" or "released" in the computer. Manufacturing does its process planning, tool design, NC programming, etc., interactively against this data using the geometric algorithms already employed in design. Quality Assurance, using computerized inspection, interacts with the design data base to determine deviation from design specifications.

In a more highly developed stage of integrated interactive CAD/CAM, manufacturing, can, by accessing the design data base early on the viewing a part on a scope, evaluate such issues as cost and problems of manufacture before a design becomes finalized. In a related fashion, the designer can have access to machinability, machining processes, tooling requirements when he faces design alternatives.

We have alluded earlier to the limitations imposed upon design-manufacturing communication by the conventional use of geometry in describing a part. The three dimensional concept in the mind of the designer is reduced to the traditional set of orthographic and isometric projections--two dimensional--which require the process planner to reconstitute that 3D concept in his own mind. It may occur, indeed, that this representation is not unambiguous.

This deficiency of the traditional drawing is actually accentuated on the scope face, unless the picture is very large, for it is difficult to display simultaneously the three orthographic and one isometric projections which on a large drawing do serve to convey special relationships with some degree of effectiveness. 3D graphics would help alleviate this problem; otherwise, some of the gain to be realized in interactive N/C programming is lost.

Mini-computer-based IGDS systems have some temporary advantages over large scale integrated interactive systems in respect to design/manufacturing communication arising out of the interactive drafting facility itself, which should be offset to some extent in the near future by effective geometric modeling/drafting software for large-scale computers.

One aspect of design/manufacturing communication calling for a fresh new approach in view of the computerization of the design/drafting process is the handling of dimensions and tolerances. At Pratt & Whitney Aircraft we find ourselves confronted by a variety of tolerancing problems arising out of the transfer from our traditional unbounded geometry to the bounded geometry of the IGDS. Beyond these, however, in consideration of the computing facilities now available, an effective mathematics of tolerances is needed to replace the tolerance manipulation from gage point to gage point which the process planner or tool designer finds necessary.

The IPAD project, concerned initially with the requirements of integrated interactive computer-aided design and with not much more than a glance at manufacturing requirements, has turned to consider CAM seriously, for the software under development is as applicable to CAM as it is to CAD. IPAD is currently planning to set up, in collaboration with ICAM, through the COCAM board, a joint ITAB/COCAM workshop on Problems of Engineering/Manufacturing Communication for late 1980 or early 1981. This should go far as a first step to establish the present status of this aspect of CAD/CAM together with the various routes along which the technology is developing.

We should like to close this discussion with some remarks about the difficulties of establishing an integrated system apart from the obstacles imposed by these technology problems.

There are presently available sufficiently effective hardware and software to carry an integrated interactive CAD/CAM system a considerable distance, and the technology gaps cited above are not severe enough to block the development of a fairly effective system. There are, however, other resources which are essential to such a development.

(a) a top-rate computer systems staff.

(b) a top-rate applications programming staff.

(c) engineers with vision and perspective in both design and manufacturing engineering.

(d) management support.

(e) a catalyst (an individual or small team) which will get the process going and continue to sustain it.

We would note also the need for guidance on the part of a company seeking to establish an integrated interactive CAD/CAM system. For many of those companies represented on ITAB or seated as observers at ITAB meetings, the concept has become steadily clearer over the past four years. Because this technology is still young, there are very few organizations in a position to advise and provide guidance. Some companies like A. D. Little, Inc., are attempting to do this, and perhaps the computer vendors or software houses will step up to this problem of assisting with such guidance and leadership--when they have fully accepted the concept themselves.

REFERENCES

1. E. N. Nilson, "IPAD Usage and Introduction into Engine and Component Design", AIAA/ASME 19th SDM Conference, Washington, D.C., Apr. 3-5, 1978.

2. E. N. Nilson, "Technical Data Base Requirements in Engineering and Manufacturing", ICAM/IPAD Data Base Workshop, Dallas, Texas, Apr. 24-25, 1979.

3. E. N. Nilson, "Integrated Interactive CAD/CAM", Israel Society for Computer-Aided Design and Manufacturing, Beersheba, Israel, Dec. 27, 1979.

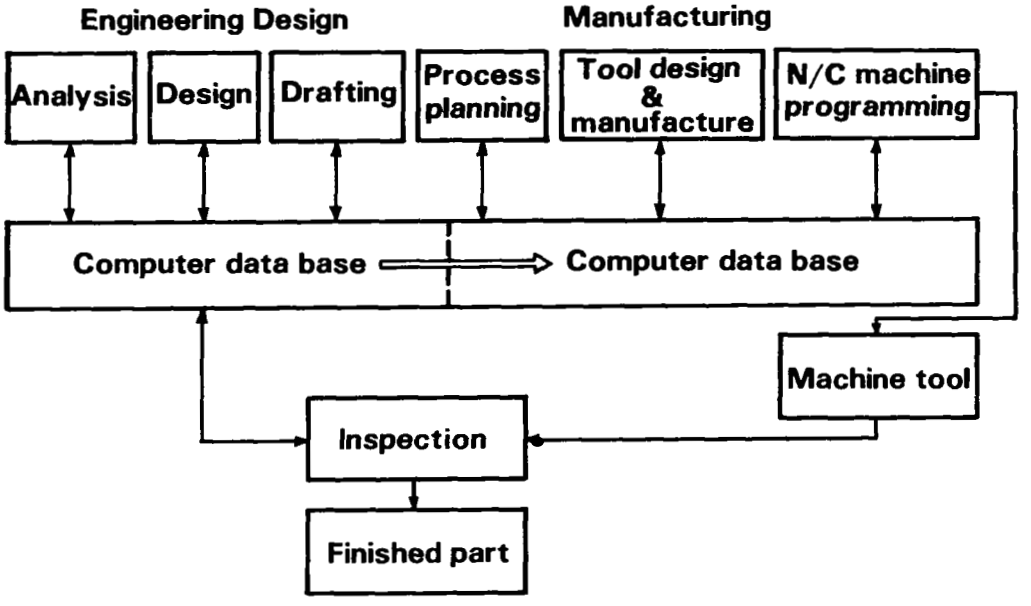


Figure 1.- Interactive Computer-Aided Design/Computer-Aided Manufacture (CAD/CAM).

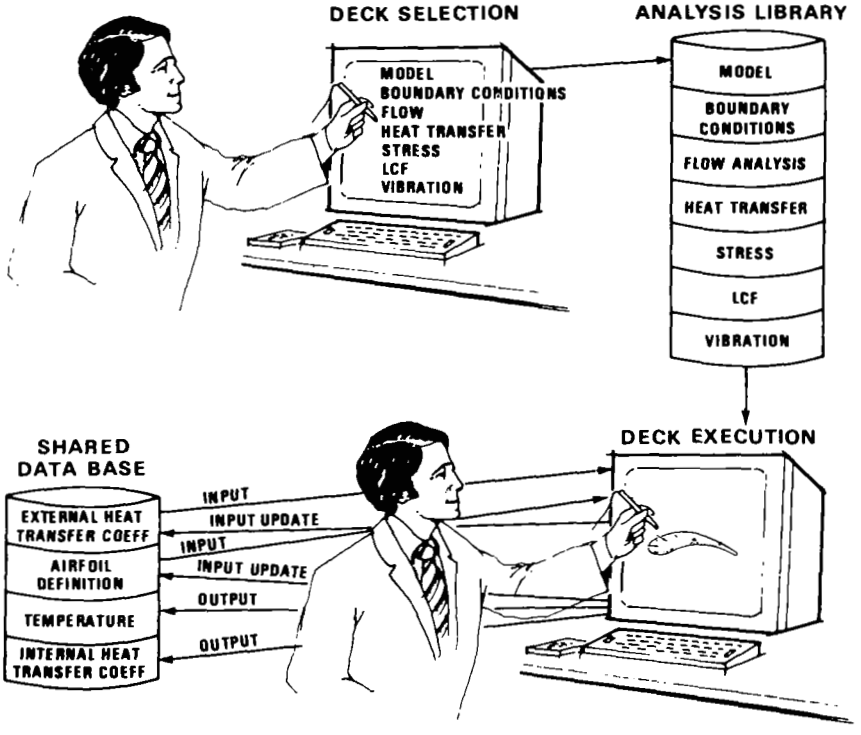


Figure 2.- Interactive design system operation.

COMPUTER-AIDED DESIGN DEVELOPMENT

TRANSITION FOR IPAD ENVIRONMENT

Herschel G. Owens
Manager, Computer-Aided Design
North American Aircraft Division
Rockwell International

Wendell D. Mock
Supervisor, Structures Systems
North American Aircraft Division
Rockwell International

James C. Mitchell
Program Manager, CAD/CAM Systems
Information Systems Center
Rockwell International

SUMMARY

Nearly a decade ago, Rockwell International's (Rockwell) North American Aircraft Division (NAAD) established a long-range development plan for integrating applications software used in the aircraft design process. These plans have been continually revised as a function of computer technology emergence, business environment, productivity requirements, and competition in the aerospace industry. The overall applications program integration activities and planning have relied upon the NASA-Boeing Integrated Programs for Aerospace-vehicle Design (IPAD) development to produce the necessary software for the data base management system and the executive controller. This paper contains a discussion of the relationship of federally sponsored computer-aided design/computer-aided manufacturing (CAD/CAM) programs to the aircraft life cycle design process, an overview of NAAD's CAD development program, an evaluation of the CAD design process, a discussion of the current computing environment within which NAAD is developing its CAD system, some of the advantages/disadvantages of the NAAD-IPAD approach, and CAD developments during transition into the IPAD system.

INTRODUCTION

CAD/CAM Requirements

United States industry will be faced with a serious productivity problem in the 1980's. Industrial competitors such as Germany and Japan are improving the industrial productivity output of their manpower by more than 4 percent per year. At the same time, productivity in the United States is declining. The aerospace industry and other industrial concerns already have been and will continue to be faced with a situation where competition for skilled manpower will increase. For example, next year 100,000 engineering positions will be competing for 70,000 graduates from our engineering institutions.

An obvious solution to this problem is to increase the productivity of the current work force to provide lower product cost, decreased manpower requirements, and reduced time in designing, manufacturing, and testing of products. The key to improvement is the intelligent automation of design, manufacturing, fabrication, and testing tasks. The technology areas of CAD, CAM, and computer-aided test (CAT), and the proper tools available to support these technology areas, are central to productivity improvement. Although this paper primarily addresses CAD, the interaction to CAM and CAT are critical parameters to be considered in the overall integrated automated efforts.

It is appropriate to consider how and where CAD developmental programs fit into the overall acquisition process for major defense systems. Figure 1 displays the major acquisition milestones and correlates the related defense system development activities, managerial functions, and relationship that major Government-sponsored CAD developmental programs have within the overall acquisition process. Note that major defense system acquisition proceeds in stepwise phases and therefore sets the stage for the tools development process. The proposed Air Force ICAD program is more oriented toward activities associated with program initiation (DSARC Milestone 0) and demonstration and validation (DSARC Milestone I) than is the Air Force's ICAM program. Both programs could appropriately be supported by objectives set forth by NASA's IPAD program. In addition to these relationships as the major defense system proceeds through its evolutionary cycle, there also evolves a requirement to involve more and more depth of design and analysis. This requirement dictates a need for computer-aided systems that are responsive to a given design phase that a particular system is currently entering or moving toward. Figure 2 presents the disciplinary and technical skills associated with an aircraft system as it proceeds into each of the design phases. Not shown are the numerous functional groups that must be part of the full-scale development, production, and

deployment activities. Although this paper emphasizes CAD activities, consideration of design manufacturing interactions is necessary to properly develop CAD/CAM systems responsive to the aircraft system acquisition process.

CAD is considered to be an integrated process embodying the complete flow of computerized data, from conceptual design through the completed engineering drawings and the equivalent sets of stored data that describe the product to manufacturing system users. The basic goal of CAD is the development of a computerized design process to significantly reduce labor costs, reduce calendar time, and to ensure that the system is viable and ready for production application. The ultimate objective is the development of a full-scale and totally integrated CAD system to support the entire aircraft life cycle design process, including aircraft design, production, deployment, and operations.

NAAD Background

NAAD has been intimately involved in the development of CAD techniques for the conceptual, preliminary, and detail design of aircraft systems. This development activity, initiated in the early 1970's, was organized to utilize existing analytical and newly developed modules, within a framework of support to disciplinary functions within NAAD. At the same time, new developments in interactive graphics, distributed computer systems, and analytical procedures were continuously being integrated into the planned CAD activity. This paper discusses NAAD's plan for integrated aircraft design, its evolutionary development, and its relationship to the NASA IPAD program.

In the early 1970's, Rockwell's computing technology had evolved from computing systems at divisional levels to a centralized corporate computing center concept. All of the main frame systems which serviced NAAD were at Rockwell's Information System Center, at Seal Beach, California, with various Rockwell divisions being serviced by a distributed network. Other centers serviced divisions in the Midwest, East, and South. By the mid-1970's, advancements in minicomputers and telecommunications made distributed processing an attractive approach, with interactive graphic packages placed on the local processors, communications protocol established with the main frame hosts, and extensive computing allocated to the main frames. This was the computing environment in which NAAD evolved its CAD plans.

Application software, to help analyze aircraft designs, had been developed or was being developed to satisfy the technical requirements of each functional group in aircraft design. As this software became more sophisticated in analytical depth, the size of input and output data grew accordingly. Initial

efforts to integrate analysis of separate software modules resulted in stand-alone preprocessors and postprocessors to enable data transfer between modules.

Until the late 1970's, computer costs were regarded by NAAD management as a high-expense item. However, the trend of decreasing computer costs, for both hardware and processing units, coupled with increasing engineering rates, had become apparent to NAAD management. Accordingly, more emphasis was placed on integrating the entire aircraft design process to improve engineering productivity, reduce design throughput time, and improve quality of designs.

It was in this environment of computing hardware and software capabilities that NAAD made a thorough evaluation of the increasing complexities, costs, and schedules required for aircraft system design. The aircraft system design life cycle which governed these considerations is shown in figure 3. The increased use of technically oriented computer programs required extensive manual processing of data to transfer data to other programs during the synthesis or analysis of the design. Emerging from this study was a long-range plan for the integration of application programs used in the aircraft design process. This continually updated development plan considered the NASA IPAD program which originated from the same common requirement to integrate the application programs for improved productivity.

The NAAD CAD plan recognized that the IPAD development would produce a data management system and an executive controller as well as other user functions. With this in mind, the approved CAD plan directed the development effort toward the evaluation and implementation of available CAD modules, where possible. In those areas where suitable application programs were lacking, NAAD proceeded to develop the computerized capabilities. These CAD modules were being integrated through a series of preprocessors and postprocessors developed specifically for aircraft system design. These preprocessors and postprocessors were designed for easy modification to the IPAD format so that NAAD could utilize the technical data management system and executive control system being developed by the NASA-Boeing IPAD team.

Rockwell was an early entry into CAD analytical tools for design of aircraft, space vehicles, and large boosters. In addition to in-house development of these capabilities, Rockwell has supported the IPAD program through active participation in the Industry Technical Advisory Board (ITAB). Rockwell has conducted a continuous review of IPAD technical developments, starting with the initial feasibility studies. Some of the NAAD CAD planning effort and study activities were found to parallel the IPAD-conducted studies with similar results. The aircraft design life cycle (figure 3) is similar to that identified by IPAD. Critical problem areas were identified such as data management, executive control, and front-end geometric data processing, which were also similar to IPAD findings.

EVALUATION OF THE CAD PROCESS

A number of major engineering disciplines, as shown in figure 2, are involved in the design of aircraft systems. These disciplines include those involved in conceptual, preliminary, and detail design activities. Critical to all design tasks is the capture of three-dimensional geometry and the evolution of master lines and dimensions for the vehicle. This task is critical for the configuration design process illustrated in figure 3 and design levels II through V for the verification of the selected configuration. Technical disciplines such as configuration development, vehicle synthesis, aerodynamics, propulsion, performance and control, etc. (figure 2), all require computer-aided tools to conduct their design task.

Many of the available tools were in the form of batch processed computer programs with little, if any, interactive capability. Since the order of required design cycles was never known for certain, the easiest approach was to handle each computer program (module) in a somewhat independent fashion. This led to the development of preprocessors and postprocessors of data to enable linkage of modules. Typical modules are summarized in table I.

Data derived from the preprocessors were listed in files in standard 80-column format for input to the next module. Postprocessors data were listed in similar files except that various output formats were organized for data printout and for batch-type graphics display. During the 1975-78 time frame, a significant amount of study was devoted to applying data base management techniques to handling these data. A number of data base management systems (DBMS) which were investigated for their usefulness included IMS, TOTAL, INQUIRE, and ADABAS. Use of the business-oriented DMBS's did not meet engineering requirements due to the complex organization of the engineering data, the size of the data base, and the inherent limitations of the DBMS. At the same time, command and control syntax for a "friendly" user interface was investigated. These user functions resulted from requirements of the technical disciplines, use of technical modules in some type of sequence, intermediate display of decision data, and network organization of the design process.

It became apparent to NAAD investigators that a new DBMS approach was required to provide an "envelope" in which to insert the technical modules, and that this "envelope" must include executive control syntax, data base management, and versatile user functions. IPAD developments during the latter part of this period were being directed with emphasis on development of IPIP (the IPAD information processor), IPEX (the IPAD executive control), and IPAD user functions and interfaces. NAAD's CAD plan was then based upon the successful development of these IPAD capabilities which could help satisfy NAAD CAD system requirements.

NAAD established some general goals that were required to remain competitive in the aircraft industry. During the conceptual design phase, it was determined that NAAD needed to evaluate 10 times as many concepts during the same allocated period of time. In the preliminary design phase, the depth of the analysis and definition of design characteristics must be improved for the same cost and schedule constraints. In the detailed design phase, the engineering cost and schedule must be reduced to improve productivity. These three phases are shown in figure 4 where virtually thousands of concepts must be evaluated during conceptual design. A few configurations are evaluated in some depth during preliminary design. During the detail design phase, a single configuration is engineered in great depth to produce a product that can be fabricated economically.

Similar to the early IPAD tasks, NAAD delineated the aircraft design process, defined the system requirements, identified the software/hardware needed, and conducted evaluations/development of application programs to support the design process. The Rockwell description of the design process was similar to the IPAD definition. Similar to IPAD, Rockwell chose to utilize locally installed minicomputer systems in conjunction with its large IBM, CDC, and Univac mainframes centrally located at the Information Systems Center in Seal Beach. Figure 5 schematically displays the distributed computing system network philosophy currently being employed.

Since the NASA-Boeing IPAD development of the data base management system and executive controller was underway, NAAD proceeded to use file management techniques and technical module preprocessors to circumvent the data base management system that was expected to be available from the IPAD development.

Results of the aircraft design process delineation tasks (figure 4) identified a number of CAD functions for the design activity levels involving conceptual, preliminary, and detail design. These functions were identified as critical elements in developing the NAAD CAD capability (i.e., design retrieval, design, geometric modeling, synthesis/analysis, etc). (See figure 6.) Concentration of NAAD resources were then placed into the following areas:

(1) Synthesis and analysis

 (a) Vehicle sizing

 (b) Aerodynamics/fluid dynamics/thermodynamics

 (c) Vehicle performance

 (d) Loads/structures/mass properties

(2) Geometric modeling

(a) Configuration development

(b) Internal/external packaging

(c) Finite elements

(d) Wiring schematics

(3) Design and drafting

(a) Two-dimensional mechanical design

(b) Three-dimensional mechanical design

(c) Orthographic and isometric drawing

This approach, along with developments in computer technology and CAD/CAM application technology, established a particular development path for NAAD. Tasks requirements were to accomplish the following:

(1) Defer expensive development of DBMS techniques to IPAD and concentrate on using file management techniques

(2) Bring on-line, as soon as possible, specialized interactive graphics capability to support configuration development, aerodynamic analysis, vehicle lofting, element modeling, and interfaces to structural analysis programs

(3) Install, test, and implement a two- and three-dimensional design and drafting capability to interface with the interactive graphics capability and key technical modules

(4) Apply critical developments, in a timely fashion, to ongoing projects to access their value in the CAD environment

In 1979, CAD development had evolved to the point where additional planning was required to define an approach for integrating the detail design task for structural analysis into the total concept. Over 40 technical modules required integration to satisfy the structural design and analysis task. These programs included the following capabilities:

(1) Configuration development

(2) Structural weight estimation

(3) Computer-aided lofting

(4) Flutter optimization

(5) Vibration and flutter analysis

(6) Aeroelastic tailoring

(7) Strength optimization

(8) Finite-element modeling

(9) Finite-element analysis

(10) Fracture analysis

(11) External/internal loads

(12) Mass modeling

(13) Stiffness modeling

(14) Others

These modules were on dissimilar computers using file management techniques for data management. The structural plan increased the demand for IPAD-type capability at the division (i.e., data management, executive control, user functions).

COMPUTING ENVIRONMENT

Rockwell was an early entrant into interactive graphics. However, costly computing resources required to drive the refresh terminals tended to discourage extended use. The early CAD/CAM computing environment is illustrated in figure 7. Other companies (namely, Lockheed and McDonnell Douglas) continued the development of CAD software to drive the IBM refresh terminal. The major breakthrough came in the early 1970's when low-cost storage terminals were introduced. NAAD interactive graphics activity followed close by the storage tube technology.

At the same time, a number of vendors were supplying turnkey systems structured around minicomputers and storage terminals. The NAAD CAD plan took

advantage of these developments in concepting a hardware plan to support the integration of aircraft design technical modules. Key to this plan were the following elements:

(1) Corporate mainframe computers (IBM, CDC, Univac)

(2) Front-end processors (turnkey CAD/CAM system, specialized minicomputer)

(3) Distributed processing protocol (telecommunications, communications software)

The CAD plan initially was structured around a Sperry Univac minicomputer to provide the rapid response needed in conceptual design. During the ensuing development, a second Sperry Univac minicomputer and a six-station Computervision system for three-dimensional design was installed. In addition, six Tektronix 4014 terminals, one Megatek refresh terminal, and 18 alphanumeric/raster scan graphics CRT stations were installed. The computers are supported by tape drives, disk drives, card readers, printers, pen plotters, and electrostatic printer-plotters. A schematic of this minicomputer is shown in figure 8. Additional graphic terminals, Computervision systems, and minicomputers are in the planning process.

Key to the plan was a logical distribution of functions between the distributed node at NAAD and the more powerful computing resources at the corporate Computing Center. Limitations included restricted storage capability at the local site and communications speed and accuracy. In order to effect a proper balance, the plan identified interactive graphics as a local responsibility, with a number of these tools placed on IBM-TSO time-sharing. Where the size of the data bases permitted local storage, they were compiled on the distributed minicomputer. Otherwise, principal data bases were allocated to storage on the corporate Computing Center mass storage facilities.

The corporate Computing Center, at Seal Beach, is about 40 km (24 mi.) from the NAAD facility. The center contains four IBM 370/3033's, two IBM 370/168's, one CDC CYBER 176, and one Univac 1100-83. These mainframe hosts are supported by literally dozens of printers and disk and tape drives. IBM's latest mass storage device is available to users of the 3033's. Over 200,000 tapes are maintained to support various divisions of the corporation.

Communications between NAAD and the Center are supported by wide-band multiplexers which can provide dedicated line speeds of up to 56,000 baud. However, most direct access terminals to machine parts are driven at speeds of 9,600 baud or slower. Rockwell is now working on a telecommunications network which will provide direct satellite communications between computing centers and high-speed transmission from the centers to distributed subhosts at the divisions. NAAD is now served by a subhost which provides remote job entry access to the Center hosts.

The NAAD CAD program provided specific user requirements to the planning activity for upgrading the Corporate Computing Center. This resulted in an expansion of communication controllers for the IBM and CDC mainframes. The program also resulted in a recognition that interactive graphics computing techniques for CAD require a local subhost for adequate support.

The major problem with the distributed computing concept (figure 5) is in allocating data storage. It should be noted that the distributed network shown in figure 5 includes existing facilities (solid lines) and planned expansions (dashed lines). The planned "analysis" computers will actually be subhosts with 32-bit architecture and fairly large random-access storage capability. Data bases required for daily activity should reside on these storage devices, while security backup to all data bases, program source files, etc, should reside on the Center hosts. Local data include those associated with an ongoing project, disciplinary modules, and drawings. Data bases are classified into geometric, numerical, and textual categories, and again subdivided into additional categories associated with projects, disciplines, and individual engineers. IPAD's IPIP is aligned to handle this type of data segregation.

APPROACH

The current NAAD approach to the integration of CAD modules has been the development of preprocessors and postprocessors that reformat the data from one module to a format acceptable for the next application program. The data have been maintained through file management, with verbal or written communication notifying the subsequent user that the data are available. These specialized preprocessors and postprocessors are designed to efficiently prepare the data needed for specific application programs. Although this approach may result in less computer costs, the availability of the data is limited to those specified application programs. However, in the environment that NAAD CAD capabilities were developed, this was the most appropriate approach. The preprocessors and postprocessors developed for specific scenarios in the aircraft design process may be readily modified to reformat the data suitable for a generalized data management system such as IPAD.

The NAAD view of IPAD is that it will provide a common and consistent data base extending from conceptual, preliminary, and detail design through manufacturing. The data would be accessible in various formats by a large number of users. This data base may extend beyond a single project to provide data for several ongoing projects. NAAD's initial intent is to capture the advantages on separate projects. These advantages include improved management control of the engineering data being used in the design. It is anticipated that the aircraft systems definition manuals will be reduced to indexes of data stored, thus reducing the manual preparation of hard-copy manuals and the associated delays in releasing the data for design. The IPAD executive controller will allow the user to request data in English, rather than restrict the user to skilled programmers. The ability to ask for data in general terms reduces the probability of error as compared with current methods that require specific data requests. The IPAD system is expected to have automated communication features available to the data base administrator for notifying users of the data availability or changes made to the data. These readily available data in acceptable formats are expected to reduce labor costs and design schedule requirements, resulting in improved engineering productivity and reduced design costs. The IPAD data base is further expected to interact with the CAM system to reduce the design change impact on manufacturing.

IPAD SYSTEM TRANSITION

Rockwell corporate management views IPAD as a major technical development that may be adapted to the needs of the various divisions of the corporation. With a view toward an economical implementation and exploitation of the IPAD system and IPAD released products, NAAD has been assigned the lead division role, with the Space Systems Group and Information Systems Center the supporting role, in the adaptation of IPAD at Rockwell. Several implementation plans are under consideration. Each plan requires the full cooperation of CAD development engineers and computing center systems programmers to successfully integrate IPAD into the design system.

Initially, IPAD would be installed on the mainframe computer system at the central computing facility. The installation of IPAD and the testing and verification of the delivered capabilities would be accomplished jointly by computer systems programers and IPAD development engineers.

Secondly, installation of IPAD on a minicomputer system at the local division facilities with network communication to the large mainframe computer would be accomplished. Testing and verification of the IPAD distributed computer system capabilities would be accomplished through close cooperation with local system programmers and the centralized computer system programmers.

During the final verification tests conducted on both the mainframe IPAD system and the distributed IPAD system, evaluations will be made of the capabilities and efficiencies of each configuration. These studies are intended to provide information relative to the advantages of each configuration, and where possible, to identify regions where technical and efficiency enhancements may be made. Special considerations should be addressed to the unique features of the host computer operating system that complements the IPAD system.

Concurrent with these IPAD installation and verification tasks, preprocessors and postprocessors for proprietary programs and public domain programs not included in the IPAD release must be modified or developed. The prioritization of these tasks will vary with each company, depending on their current needs. At NAAD, the largest data base is generated through the Structural Design and Analysis System (SDAS) by the multiple technical modules it employs. Since Rockwell is an extensive user of NASTRAN, it is likely that the modification of Rockwell NASTRAN preprocessors and postprocessors would be accomplished early in the development phase. Certainly, the incorporation of a more recent version of AD2000 three-dimensional design and drafting graphics into IPAD is warranted. The development of preprocessors and postprocessors for NAAD's configuration geometry and dimensional control system of programs to provide vehicle moldline data is a high-priority task. There are other technical modules that would subsequently be integrated into Rockwell's IPAD installation.

Another mode of operation that may be explored is accessing the mainframe IPAD system via TSO or INTERCOM programs. This approach may be suitable for divisions whose facilities do not warrant a distributed IPAD system. This capability would offer these remote facilities the benefits of IPAD without incurring the capital outlay for a minicomputer.

The incorporation of IPAD into the design system is a long-term development effort that encompasses several planned phases. An IPAD configuration control board (CCB) is recommended to assure that IPAD has been tested and verified after each phase of development. The CCB would be comprised of representatives of each participating division who could provide liaison with their organization and assist in the implementation of IPAD throughout the corporation.

As noted, the initial thrust of IPAD will be in the structural design and analysis disciplines. It is expected that the aerodynamic technologies would then be incorporated. After the technical modules are functioning within the IPAD system, it would be appropriate to integrate the project management systems.

The IPAD level I system is expected to be released for use on the CDC computer systems in the spring of 1981. It is anticipated that this same

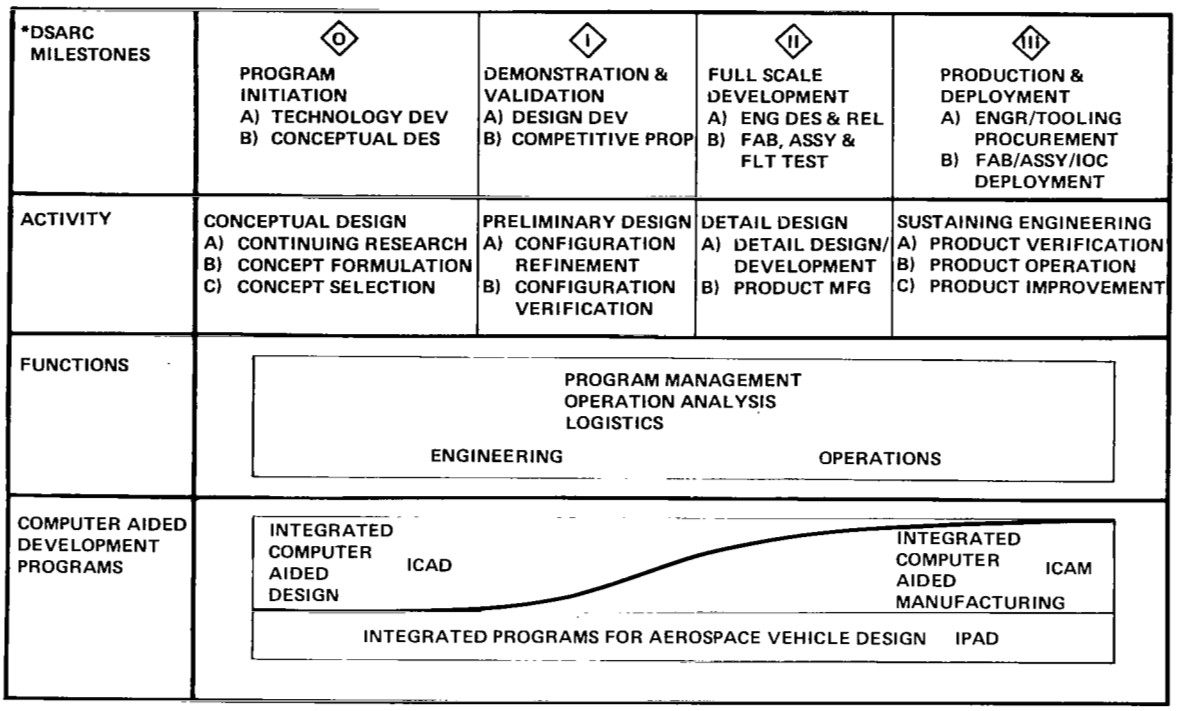
capability will be available for the IBM computer systems by the end of 1981. Although the IPAD level I releases are considered to be functional prototypes of the production systems planned for the level II and III releases, significant developments in technical data management have been completed and may be adapted for use in the near term. Additional IPAD products which will be released as the development progresses may be integrated into the CAD system in a transition phase. It is expected that the current CAD system may function independent of IPAD or in a mixed mode using the IPAD product capabilities as this transition progresses.

Certainly, the real test of the system is in the application to a major project. At Rockwell, it is anticipated that IPAD will be applied to subsets of the project prior to committing the system to the entire project. As applications for IPAD are extended to entire project designs and subsequently to multiple projects, the scope of IPAD must be extended to assist in productive interactions between engineering and manufacturing.

It is with a great deal of anticipation that Rockwell addresses the implementation of IPAD for technical data base management. It is reasonable in the near term to have IPAD operational on both CDC and IBM computer systems as stand-alone mainframe systems and as distributed computer systems with locally installed minicomputers complementing the centralized mainframe computers. These capabilities may be utilized by the various divisions of Rockwell involved in aircraft, spacecraft, propulsion, automative, telecommunications, graphic systems, and energy systems. The IPAD technical data management development may be used in the Integrated Computer-Aided Manufacturing (ICAM) and Integrated Computer-Aided Design (ICAD) projects. The appropriate combination of these capabilities into a totally integrated system is expected to provide significant productivity improvements.

TABLE I. - SUMMARY OF TYPICAL TECHNICAL MODULES

Discipline	Acronym	Program Description	Design Level
Vehicle synthesis	VSPEP	Vehicle Sizing and Performance Evaluation	II - III
	SWEEP	Structural Weight Estimation Program	III - V
	SWAIP	SWEEP/Production Cost Model Interface Program	III - V
	DWGRD	Grid/Air Vehicle Mass Properties Program	III - V
	RASCAL	Rockwell Automated System for Computer-Aided Lofting	III
Vehicle dynamics	COMET	Structural Flutter Optimization Program	III - V
	STAR 6	General Vibration and Flutter Analysis Program	III - V
Aerodynamics	FUDP	Flexible Unified Distribution Program	III - IV
External loads	ADELS	Advanced Design External Loads Program	IV - V
	XLOADS	External Structural Loads Program	III - V
Structural analysis	NASTRAN	NASA Structural Analysis Program	IV - VI
	FASTOP	Flutter and Strength Optimization Program	IV - V
	AC-5 AC-11	Honeycomb Sandwich Panel Stability Under In-Plane Biaxial Loads and In-Plane Shear Loading	IV - VI
	AC-31 AC-32 AC-33	Laminate Design for Strength and Stiffness-Multiple Load Condition, Upper Boundary Optimal Design, and Biaxial In-Plane Loading	IV - V
	AC-120	Minimum Weight Design Advanced Composite Panel Strength and Stability	IV - V
	RIFEM	Rockwell Finite-Element Model Pre and Post Processor	III - VI
	RASSP	Rockwell Automated Stress Spectrum Program for Fatigue and Fracture Mechanics	IV - V
Structural optimization	SPARS	Multispar Box Optimization Program	III - V
	AC-88	Aeroelastic Taloring Structural Sizing Program	III - IV
	TS-O	Aeroelastic Taloring Structural Optimization Program	IV - V



*DEFENSE SYSTEMS ACQUISITIONS REVIEW COUNCIL

Figure 1.- Acquisition process for major defense systems.

DESIGN PHASE	AIRCRAFT CONCEPTUAL DESIGN	AIRCRAFT PRELIMINARY DESIGN	AIRCRAFT DETAIL DESIGN
DISCIPLINARY OR TECHNICAL FUNCTION	OPERATIONS ANALYSIS CONFIGURATION DEVELOPMENT VEHICLE SYNTHESIS AERODYNAMICS PROPULSION MASS PROPERTIES	OPERATIONS ANALYSIS CONFIGURATION DEVELOPMENT VEHICLE SYNTHESIS AERODYNAMICS PROPULSION MASS PROPERTIES STRUCTURES DESIGN STRUCTURES ANALYSIS VEHICLE DYNAMICS THERMODYNAMICS MATERIALS & PROCESSES AVIONICS INTEGRATION SUBSYSTEMS DESIGN	OPERATIONS ANALYSIS MASTER DIMENSIONS CONFIGURATION MANAGEMENT PROJECT ENGINEERING AERODYNAMICS PROPULSION WEIGHT CONTROL STRUCTURES DESIGN STRUCTURES ANALYSIS STRESS VEHICLE DYNAMICS THERMODYNAMICS MATERIALS & PROCESSES SUBSYSTEMS DESIGN AVIONICS INTEGRATION ELECTRICAL ELECTRONICS SYSTEMS WEAPON SYSTEMS INTEGRATION FLIGHT CONTROL SYSTEMS CREW SYSTEMS MECHANICAL SYSTEMS DESIGN AVIONICS INST. DESIGN SUPPORT EQUIPMENT DESIGN DESIGN SUPPORT

Figure 2.- Technical skills involvement.

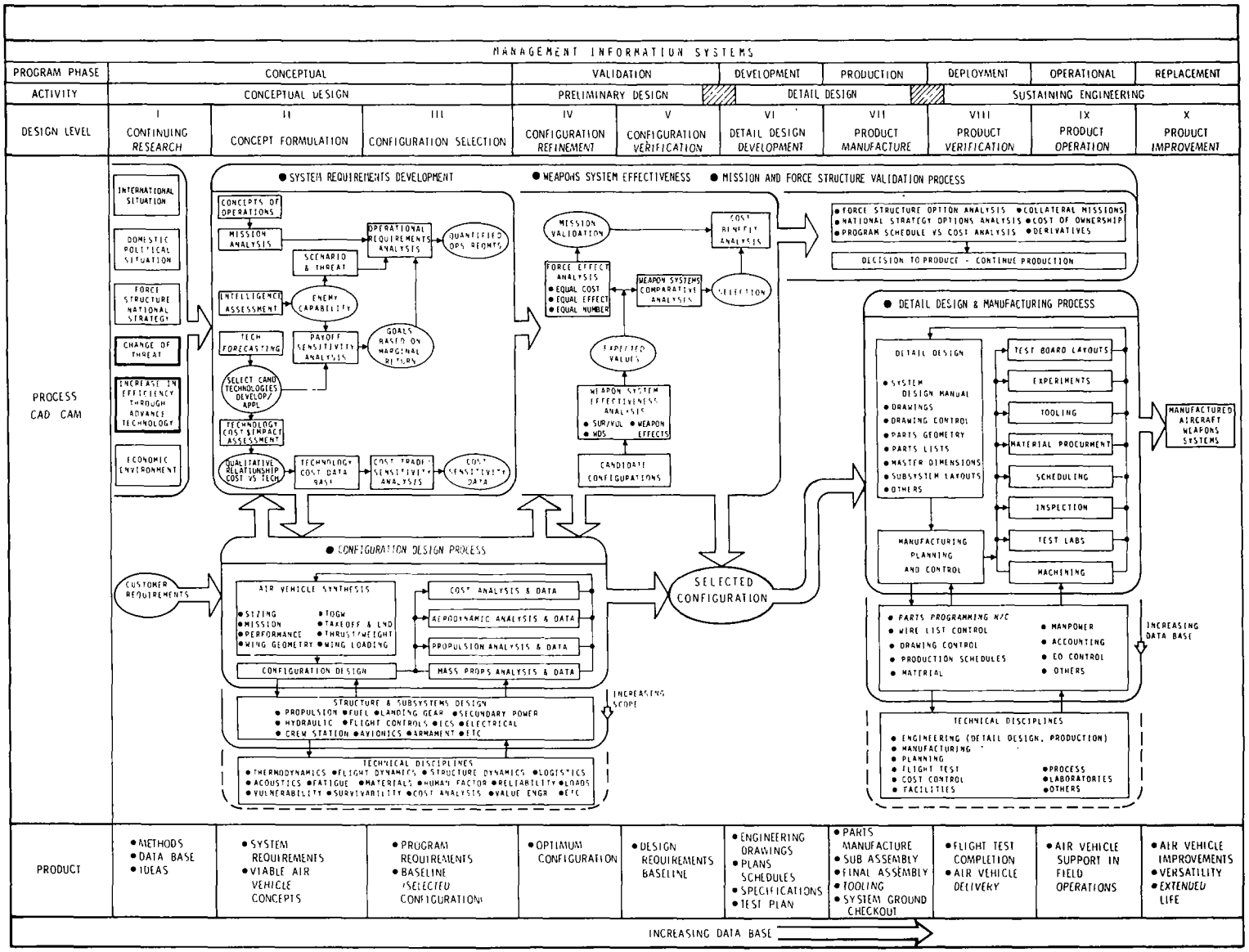


Figure 3.- Aircraft systems life cycle.

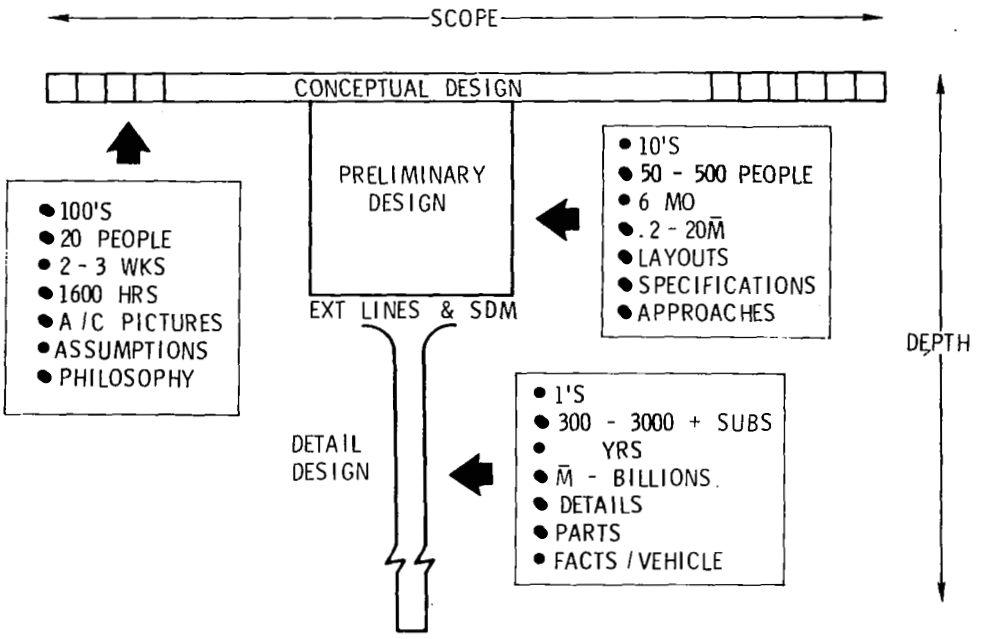


Figure 4.- Conceptual/preliminary/detail design phase characteristics.

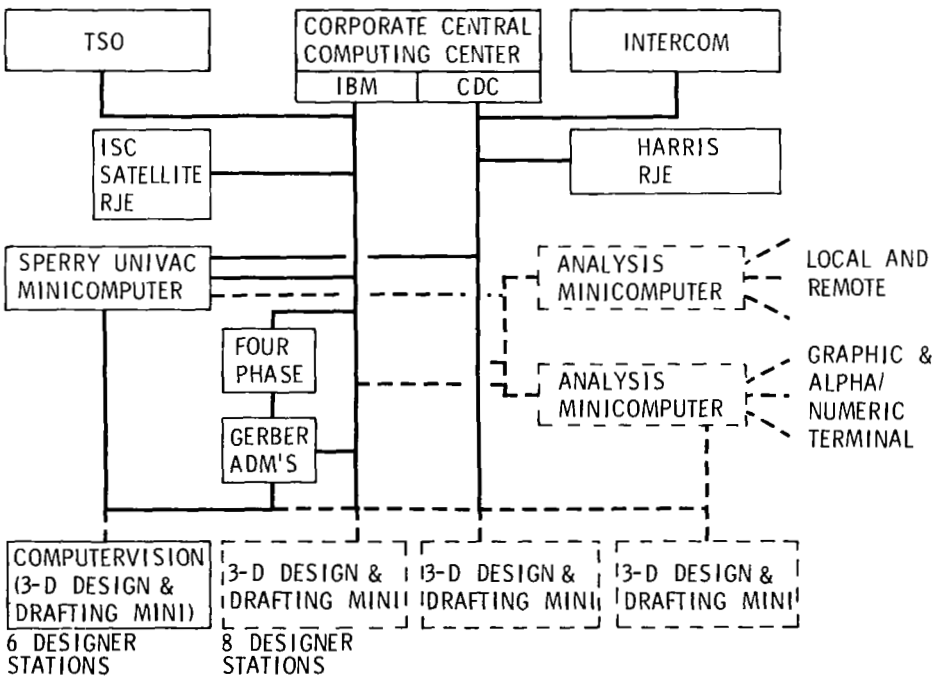


Figure 5.- Distributed computing system network.

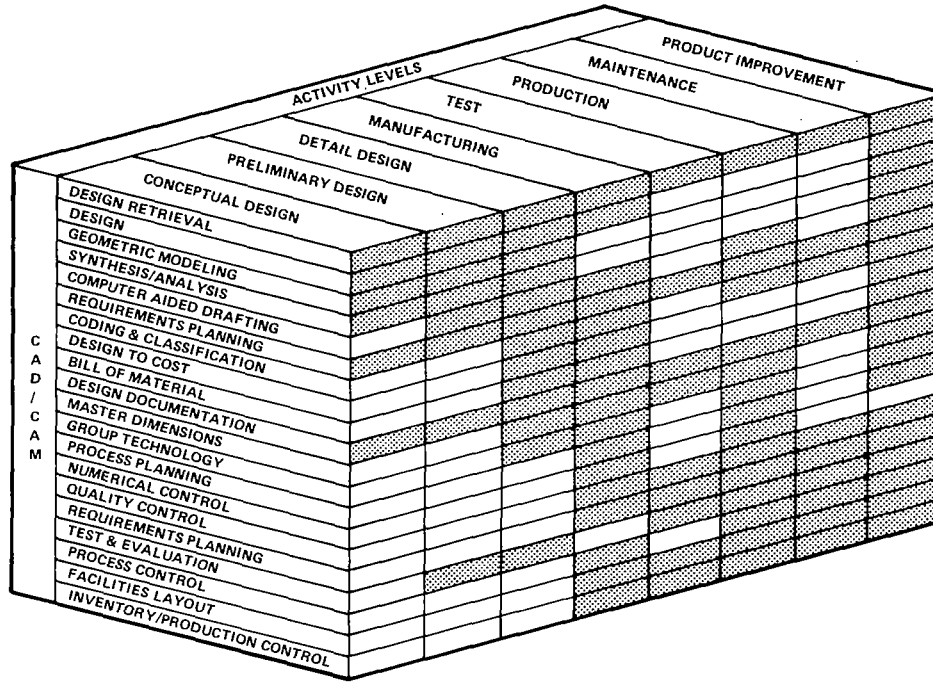


Figure 6.- Product activities and CAD/CAM functions overview.

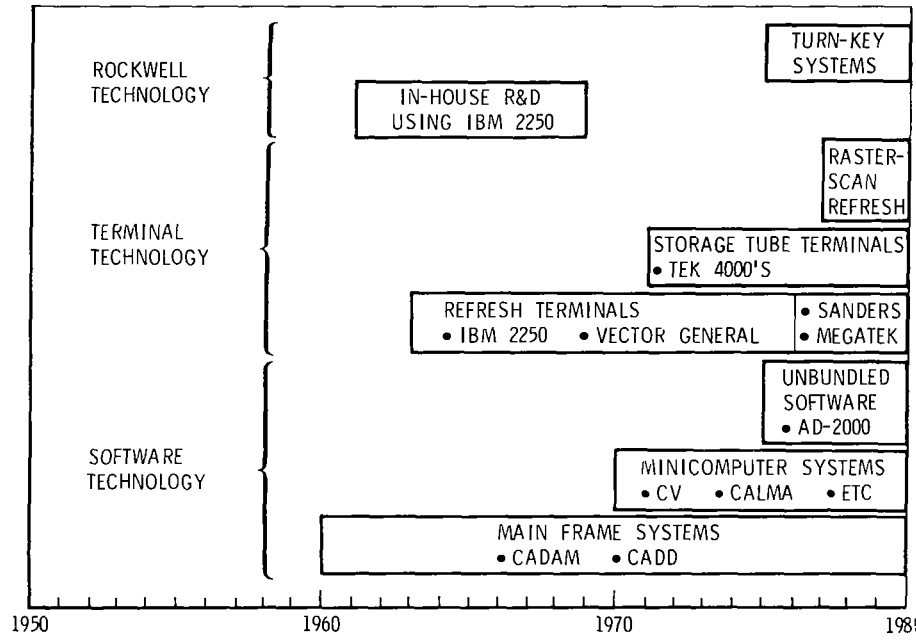


Figure 7.- CAD/CAM history overview.

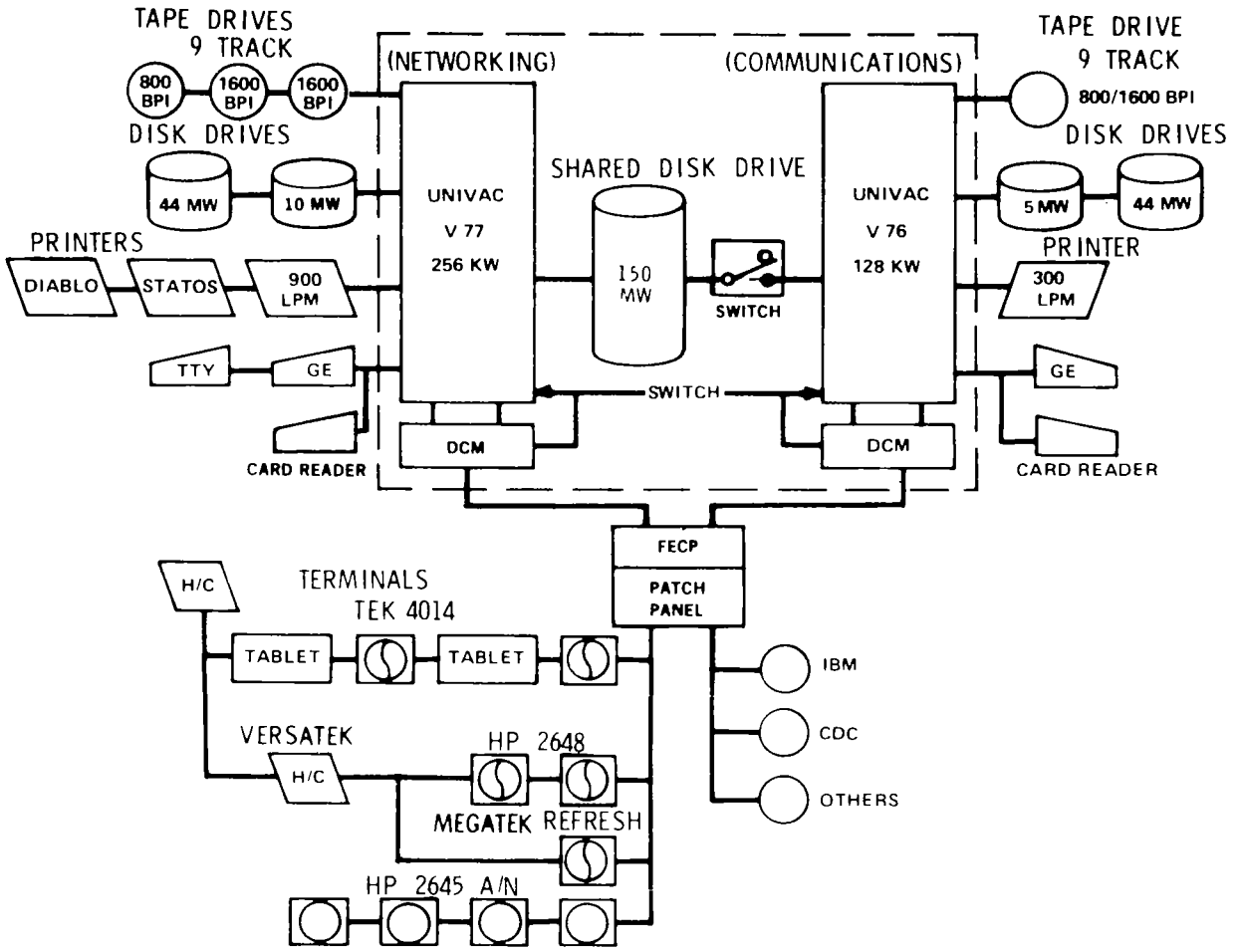


Figure 8.- Minicomputer system schematic.

IPAD: A COMPUTER VENDOR'S PERSPECTIVE

Harley D. Feldman
Control Data Corporation

Control Data Corporation has been associated with the IPAD concept since 1970. It has been our pleasure to provide knowledge, learn the technology, participate in decisions, and generally support the IPAD effort. Drs. Miller and Fulton reserve much credit for sticking to a concept as long as they have over some difficult times. Now that the IPAD concepts are reaching fruition in the engineering world, I would like to reflect somewhat on a computer vendor's view of the IPAD program as it has evolved, a perspective of the decisions made as the program moved forward, a current view of the development, and a look into the future of IPAD concepts.

Control Data became involved in the IPAD concepts in 1970. Working with NASA, Boeing, and General Dynamics, we sensed two ingredients necessary in engineering computing systems: first that the state-of-the-art of engineering and computing, at that point in time, had been unable to economically utilize the large amount of existing application software which existed in the government and industry communities and second that the key ingredient in the implementation of an integrated design and analysis system would be the monitor program to control the task flow and perform the clerical functions associated with the design process. By implementing an integrated system to take advantage of existing and future application programs, it was thought that more productive design could be achieved. The computer would become a tool to support the entire design process rather than a resource utilized by individual engineers to support individual work processes.

Many issues had to be considered when designing such an integrated system. First the scope of the system had to be defined. What product could be designed upon it? What technical disciplines should be represented? Second, analytical requirements had to be determined in terms of levels of analysis and what the flow of this analysis might be. Next system performance had to be considered to ensure that the system was usable by the appropriate engineering users. Finally requirements for computer hardware and software needed to be specified. This included topics such as utilizing pre-existing code, program and data integrity, utility functions, and management of data. These issues were presented to NASA which sought to search out some of the answers through a feasibility study approach.

During the feasibility study period, CDC was asked for technical opinions on many of the issue topics. We presented to Boeing and General Dynamics an overview of trends in hardware development and specific plans for CDC equipment. We gave opinions on the impact of hardware changes on IPAD, evaluated hardware configurations, looked at supporting existing codes, and consulted on design of a centralized data base to support engineering data. We also completed a

comprehensive study on migration of IPAD codes from the 6600 to the STAR computer to aid an analysis of IPAD and application portability between different mainframe architectures.

In addition, because of a government desire not to have operating system modifications be a part of IPAD, CDC designed two features required by IPAD design but not existent in the Network Operating System (NOS). These two features, Pause and Omit, were designed to augment existing NOS capabilities and to provide the engineer with facilities to make his job on the computer oriented toward the way he did engineering design.

Control Data then participated in a critique of the two feasibility studies. We concurred with the results which concluded that the IPAD software should be a framework under which a company's operational modules (application programs) should run. This would allow each company to customize IPAD software for its own way of doing business while taking advantage of integration technology. Our other conclusions were that (1) a single user interface language was desirable; (2) operational modules should be easily integrated into the IPAD framework; (3) the development schedules and costs were underestimated; (4) IPAD software should be as host independent as possible and yet be able to take advantage of host operating system capabilities; (5) the system must be cost effective for acceptance; and (6) it should be an extension of the design process; not a major "leap of faith" for a corporation.

There were two main recommendations to NASA from the people at the IPAD critique sessions: first that a prototype system should be developed and given to each aerospace company for evaluation in the aircraft design process and second that a computer manufacturer might be the best place to develop the IPAD framework software. NASA came to their senses by acknowledging the first recommendation and ignoring the second.

In late 1974 NASA began formation of ITAB, Industry Technical Advisory Board. This organization was to provide recommendations to the IPAD contractor on direction and priorities within the IPAD development. We elected to become members at time of formation, and ITAB has proven to be one of the most successful advisory groups ever collected to monitor a major government program.

In 1975, CDC supported the proposal efforts of both major bidders to the IPAD development contract, Boeing and McDonnell Douglas. The topics we attempted to address were as follows:

1) Operating systems - what capabilities existed then and where were they going in the future?
2) Data base management
 - o What data elements were needed?
 - o Development of a subcontractor's work statement to implement the proper data management system
 - o Study of state-of-the-art data base managers, especially relational and set-theoretic
3) Development configurations
 - o Hardware and software
 - o Availability
 - o Cost

- o Support utilities needed
- o Site management

The recommendations were to implement IPAD on a central computer, support geometry and graphics data in the data base, and develop IPAD using a higher level language.

In 1976 when Boeing was awarded the development contract for IPAD, Control Data participated in some of the early studies in support of IPAD design issues. In the data base area, IPAD requirements were mapped against our DMS-170 product for closeness of fit. We also helped Boeing design and evaluate the SUSPEND/RESTORE feature under NOS to support the task interrupt requirements. Then in 1977 Control Data exposed the IPAD staff to our three-schema data manager developments in order for Boeing to evaluate the latest in data base technology.

One final historical note was our participation in the Audit Committee in August 1978. Recommendations were made at that time to have Boeing spend less effort on a first level IPAD containing a subset of all IPAD capabilities and to concentrate on the data manager, use of geometry and graphics, and to provide a basic yet distributed executive. This approach is still in effect.

I will now assess the current IPAD technology versus the state-of-the-art in computer technology. The assessment will cover the issues of engineering data management, distributed architectures, and user interfaces.

In the area of engineering data management, IPAD specified four fundamental requirements not met by today's data management systems: engineering and scientific data types, distributed data bases, dynamic scheme definition, and extensive security and automatic recovery from failure. After reviewing the CDC data management products, we came to the same conclusion as IPAD; i.e., no commercial data manager could be used. However, our three-schema data model appeared to have the most promise. After consultations with the IPAD staff, the three-schema architecture was selected as the model from which the IPAD N-schema data manager was designed.

The main data element to be managed for mechanical design is the part geometry. Today's data managers do not understand that two points each consist of a three data element record of floating point numbers and that these points form a line in space. Therefore the geometric intelligence has to be put into each application, precisely the situation to avoid. If the data manager could supply geometric data sufficiently rich to adequately describe the physical object, much of the drudgery of writing new applications could be avoided. The data manager would provide a clean interface for the storage and retrieval of geometry.

Even without the data manager to store and retrieve geometry, a requirement exists to exchange geometry between CAD/CAM applications. IPAD had chosen the ANSI Y.14.26.1 geometry protocol to provide this geometry movement.

A current attempt at this exchange problem has been the Initial Graphics Exchange Specification (IGES). IGES specifies a neutral geometric definition

which all application systems could address and thereby communicate geometry amongst themselves. This situation does alleviate some of the problems in moving geometric descriptions from one application to another; but it also ignores others which the IPAD developments attempt to solve. While IGES presents a neutral definition of geometry, it is done at the file level, necessitating a processor to both absorb and create IGES data from each application. IPAD's geometry definition is at the lowest level, thereby requiring the simplest application calling sequence. Secondly, IGES does not have much in the way of associativity, even though this may evolve overtime. Without associativity between geometric entities, object description is inadequate.

But the main benefit to be realized from IPAD over the IGES structure is IPAD's ability to recognize changes in individual geometric entities. With IGES, an entire new file must be created to record any changes in the object geometry. This environment is not unlike the situation with drawings today. IPAD technology should take us into the next era where configuration control of the part geometry is done on the individual entity level where it would be the most efficient.

I believe that the transition from where the manufacturing industry is today to the future data management capabilities is through the IPAD concept of flat files. Flat files are those with an undefined data structure as far as IPAD formats are concerned. This environment allows the storage and retrieval of data from existing applications under control of the data manager. As the new applications are written or older applications modified to the new interfaces, structured data will then be under control of the data manager. Utilities could be written, if desired, to move data from the structured format to the unstructured, although the converse is usually not possible.

Distributed architectures have become a necessity in all areas of the computing industry because of geographical dispersion and lowering communications costs. This situation is especially true in the manufacturing area, as the engineering and manufacturing of corporate products are spread across many divisions and geographical locations. As this situation occurs, the problem of communicating information between sites in a digital form is more critical, and the management of the data becomes a necessity. Hence two problem areas must be solved: that of remote communications of human or application understandable data and distributed logical data bases supporting each geography.

In the area of communications, the computer vendors have in the past gone their own directions and have only been able to communicate to like systems. With the advent of protocols like X.25 and bit-serial channel devices the machine-to-machine communication problem has become easier. However the problem of application-to-application communication is still there, as mentioned above with geometry.

The area of distributed data bases is now getting attention because of user requirements. However, each vendor has one or more different data manager products, making the distributing of data problem very large. The IPAD design where the same logical data manager is distributed to all nodes in the network appears to be the earliest viable solution. Linking differing data managers

would require a neutral data request format resulting in mapping problems similar to those encountered with IGES.

Finally in the area of user interface, today's operating systems and applications for the most part each present the user with a different set of commands, language syntax, and output reports. Major expenditures are made to train people to move from one system or application to another. IPAD's attempt to build a single user interface is a major step forward in alleviating this expensive problem. Only when standard interfaces are defined and developed will this problem be solved.

As to the impact of IPAD on Control Data, we recognized the CAD/CAM marketplace as a viable business opportunity. IPAD has provided guidance in many difficult areas not understood by computer technicians. Rather than develop CAD/CAM oriented products in a vacuum, our involvement with IPAD has provided valuable information and requirements to aid us in developing viable solutions to part of the engineering or manufacturing problem.

As to the future of CAD/CAM technology, I feel that the progress to date is only scratching the surface of the full potential in computers supporting engineering and manufacturing to increase productivity. I can visualize major advances in several key areas including those being addressed by IPAD today.

One of these areas of impact is in definition of the product model. The CAD/CAM systems utilized today, including AD-2000 within IPAD, use two- or three-dimensional wire frame geometry and create two-dimensional wire frame drawings. These systems essentially automate the drawing process which is in practice with draftsmen today. Drawings can be generated and modified at a faster rate than a piece of drawing paper. However these systems do not help the designer visualize his designed product in a three-dimensional solid sense; they cannot represent inside or outside points to a surface, etc.

Once the product geometry is defined, the next logical step would be to communicate the definition to other applications, e.g., structural analysis and numerical control machining. But each application has a different definition of geometry built into its data structures. So constant messaging of the geometry is done by each new application or a neutral geometry definition is used (a la IGES) to which each application can communicate. Even under the neutral definition, the geometry communicated may not be rich enough in information to be useful to the receiving application.

The research being done today in the area of three-dimensional modeling systems will provide the vehicle for solving these problems. The engineer will define his product in three-dimensional simplistic solid primitives. He will work in a medium for which visualization and accurate definition are possible. The geometry definition will most likely be calculable from the topological description made by the user.

Once the geometry exists, then the other application areas can be appended directly to this definition. Because of the solid nature of the geometry definition, its information content will be sufficiently rich to

support any application area whether existing applications are interfaced or new areas are written. Eventually only new applications will exist, as they can take direct advantage of the richness of the product geometry. Therefore the lessons learned by IPAD in the area of geometry definition and management will be extremely valuable.

Another area of extreme impact will be in the engineering/manufacturing interface. Today the "fence" between engineering and manufacturing is not only counterproductive but results in extra cost and time during the product development cycle. As was concluded during the recent IPAD Workshop on Engineering/Manufacturing Interface, this interface must become much less formal and more interactions must take place. However, these interactions will necessitate changes in organizational structures, the way companies do business, and the computer technology required to support the processes.

In the computer area, data will need to be communicated from engineering to manufacturing, e.g., part geometry and materials properties. Also data will flow from manufacturing to engineering, e.g., manufacturing costs and tooling data. Engineering and manufacturing applications will need to communicate, store, and retrieve data from a common repository. This will dictate the use of a common data base manager across engineering and manufacturing. Each area will need a different view of the data base, but the data base must be sufficiently rich to support all applications. This situation presents a healthy challenge to the developers of data base management systems. The IPAD and ICAM programs will go a long way in developing requirements for the vendors of these systems.

It is possible to visualize at some date in the future where the entire product development cycle becomes somewhat automatic. The design engineer would define his product with three-dimensional visualization, the geometry would be communicated to the proper analysis programs, results returned, design changes made, the part descriptions communicated to the manufacturing engineer, and the parts manufactured by direct numerical control. The whole process from design to manufacture would be modeled in the computer prior to manufacturing the first part. Assurance of fit and tolerance adherence would occur because of the extensive modeling performed.

Much as this may seem far-out today, I believe that this result is desirable and achievable. It will not be reached in one revolutionary way; rather, by evaluation in a piecemeal fashion over time. I believe that IPAD especially has shined the light in the proper direction to guide manufacturing companies and the computer industry down the correct path to achieve the desired end. Control Data is proud to have been involved with IPAD over the past ten years and continues to support this outstanding program.

IMPLEMENTING AN INTEGRATED ENGINEERING DATA BASE SYSTEM A DEVELOPER'S EXPERIENCE AND THE APPLICATION TO IPAD

Everett A. Bruce

Digital Equipment Corporation

1. Introduction

The software developed by the IPAD project will provide a new and very powerful tool for the implementation of integrated Computer Aided Design (CAD) systems in the aerospace engineering community. It must be kept in mind, however, that the IPAD software is only a tool and, as such, can be well applied or misapplied in any particular environment. The many benefits of an integrated CAD system are well documented, but there are few such systems in existence, especially in the mechanical engineering disciplines, and therefore little available experience to guide the implementor.

Electronics design, because its object space is generally more constrained and better defined, has generally had more advanced CAD tools than mechanical design. Thus one finds more examples of integrated electronic CAD systems currently in use. Although the mechanical and electronic disciplines themselves may be quite different, the fundamental issues of implementing an integrated CAD system are very similar. This paper will present a set of maxims derived from experience in implementing an integrated electronic CAD system at Digital Equipment Corportion. We believe these maxims apply equally well to anyone contemplating an integrated CAD system built around IPAD.

2. Background

In 1973 the CAD systems Engineering Group at Digital Equipment began the design and implementation of the Integrated Design and Engineering Automation (IDEA) System, an integrated electronic CAD system for use within Digital's own engineering groups. At that time Digital had in use a number of computer aided design and analysis tools, each of which was independent or coupled to other tools by way of special purpose transfer files. IDEA was designed to replace this loosely coupled and uncontrolled set of applications with an integrated and contolled hardware/software system for CAD. The hardware environment included a network of DECSystem-10 time-shared computers, each of which supported a number of alphanumeric and GT62 refresh graphics terminals. The basic software system included a general purpose device independent graphics system, a data base for electronic design implemented using DBMS-10, a CODASYL data base manager, and the necessary data base support utitilies.

The design of the data base and the implementation of the data base support software were the major activities in the IDEA project. The data base itself consisted of four major components:

(1) A large parts library (>30,000 parts)

(2) A library of design standard shapes and parameters

(3) A set of electronic design modules containing all electrical, logical and physical data for a part, stored in a complex network data structure

(4) Directory and access control information for the entire data base.

The data base utilities consisted of a set of sub-routines for accessing the data base and programs for maintaining and updating the libraries and directories.

On top of the IDEA foundations of graphics and data base a set of electronic design and analysis programs were implemented, each of which drew its input data from the IDEA Data Base or provided output to it. These applications included logic design input, printed circuit board data entry and layout, parts library input and maintenence and manufacturing data output. Some of these applications manipulated the data base directly, while others interfaced via an extract/update file mechanism. In the latter case, system integrity was assured by always requiring an extract/update pair and not allowing another update of a given design module until the first update was performed. Applications users spanned the range from engineers to technicians to drafters.

As with many large development projects, IDEA took somewhat longer and delivered somewhat less functionality than originally planned. IDEA began user testing in 1976 and went into production use in 1977. The system has continued to expand and evolve since then, based on user feedback, technology needs and the developers' increasing awareness of the system's strengths and weaknesses. Although at times difficult, this system evolution has been valuable in both making IDEA more effective and also focussing on the techniques needed to design and implement future integrated CAD tools.

3. Some Maxims

The process to evolve the IDEA System has been both very difficult and very enlightening. Many of the problems encountered in IDEA originally surfaced as specific user issues of an apparent minor nature or as unquantified user discomforts. Given the broader perspective of time and experience, however, these individual problems coalesced into a relatively small number of global systems design issues. We refer to 'systems' here in

the normal EDP sense, that is, a collection of interrelated hardware, software, data, processes and users. A fundamental premise is that systems analysis and design are very different disciplines from hardware or software design. Problems have tended to arise in those areas where the developers were not sufficiently cognizant of system issues. On the other hand, those areas in which the systems aspects were specifically taken into account were among the most successful parts of IDEA.

The following maxims are derived from reflection on the IDEA experience. While they may appear to be somewhat obvious, many of them were overlooked or not considered sufficiently important by the tool-oriented developers of the system. They are presented here as points on which those contemplating the development of an IPAD system might reflect. A few minutes of such reflection may save many painful hours of correction.

1) Never attempt to automate that which you don't understand how to do manually.

This is probably the fundamental rule of all systems design and yet one which is frequently violated in the design of CAD systems. Most CAD applications in companies start out as sets of unrelated tools created to solve specific design problems. As the collection of tools gets large enough, the implementation of an integrated CAD system may be attempted. But the tools themselves form only a small part of the system. Far more important are the processes, the way the tools fit together, and the data with which the tools and processes operate. Far too few engineering organizations have a complete understanding of their own process flows. Typically much of the process is informal and undocumented. Fewer organizations still are in a position to quantize those processes into an integrated design system. Most engineering data today exists in a myriad of forms, much of it analog (e.g. drawings, purchase specs, models). In order to create an integrated CAD system, this data must be quantized, and even before that it must be explicitly identified.

The major design efforts of the IDEA System were the identification of all of the information about an electronic design required to form a complete data base and establishment of the appropriate relationships among those data items. Far less time was spent understanding the engineering process. This led to a few situations where the data and the process were not in phase. Users either did not have access to needed data or could get it only with great difficulty. For example, during printed circuit board layout, a user could change a feed-through hole size only by putting the design back into the data base and invoking the standard shapes editor. This was a very slow process for what appeared to the user to be a trivial change.

CAD developers have traditionally been oriented toward tools, technology and techniques. CAD tools automate a particular step in a design process. To build a successful integrated CAD system, one must look beyond the individual steps and build a system to aid the overall engineering

process. To do this, one first has to understand that process. Probably the greatest benefit to an organization which arises from an integrated CAD system project is this understanding. Even if the system were never actually implemented, great benefit would accrue to any engineering organization which made the effort to understand and quantize its own processes and data. On the other hand, any attempt to integrate a set of CAD tools without this understanding will probably cause more harm than good.

2) The system users are your most important resource.

Too often CAD system developers, as tool-oriented people, focus all their attention on the technological problems to be solved (how to define 3D geometry, how to structure the data, etc.) and forget the fact that to be successful their system must be used. But CAD systems, as all DP systems, must start with a set of users. Indeed, one of the most important first steps in the system design is to explicitly identify who the user community will be. Only when this is done will the systems' purpose and direction be clear.

The user community is the ultimate source of information about both the engineering process and the requisite data. The users should be involved from the inception of any CAD system design as both consultants and reviewers. The IPAD project at Boeing has shown the way in this regard. The engineering group is separate from the system design group and has been instrumental in aiding and validating the system design through the development of a set of scenarios and demonstrations. Companies planning an IPAD implementation should plan a similar activity. Although the Boeing effort points the way, each company will need to tailor IPAD for its own environment and the people to do this are the system's users, not the developers.

Whether an integrated CAD system is successful ultimately depends, of course, on whether it is used, and that in turn depends on whether it is acceptable to its user community. The users then must be the final review authority for both system design and individual application interface design. It is also very important that the users and developers negotiate system and application performance goals early in the design process. Performance is every bit as critical as functionality to system success. IDEA System performance goals were never explicitly stated. Users' expectations were higher than the developers' and this became a constant source of contention after the system was released.

When identifying system users, it is important to distinguish between those who supply data and those who use it or benefit from it. The former are obviously vital to the overall system success, but unless they are brought into the process at a very early stage, they may fail to see the value of the system. Indeed, an integrated CAD System may add to the work of such a group without producing any direct benefit to them. An example of this could be a purchase specification group or components engineering group

which has the data needed to populate a standard parts library. Incorporating such a group into the system after it is designed and implemented can be very difficult and frustrating.

3) Data base administration is the key to successful implementation.

Most literature on data base systems describes a function called the Data Base Administrator (DBA). The DBA is usually described as being responsible for understanding the data needs of the corporation, defining the relationships among the data, structuring the data bases and then overseeing the day-to-day operations of the data base system. In almost any significant data base system this function will be too large and demand too great a variety of skills for one individual. However, it is very important to the success of the system that the function be recognized by the corporation and performed by a set of people explicitly chartered for, and dedicated to, data base administration.

Minimal DBA functions will always be performed, whether recognized or not. When one is going to build a data base, the data it will contain and the data relationships must be defined. This is often done by system implementors. In the IDEA System, for example, the person who researched the data needs also defined the data base schemas and led the software engineering team which developed the data base software. Frequently the other DBA role, that of dealing with daily operations, is assigned to the EDP operations staff as another of their routine functions.

The danger of this implicit approach to data base administration is that it tends to focus on the details and overlook the total system. It is the data base which integrates an integrated CAD System. The high level design of that data base must be done by people with an overview of the corporation's business, data needs and processes. Too often the CAD developers lack this overview and deal only with the data needs of individual CAD tools. CAD developers also generally lack the visibility and "clout" with top management to organize the data resources and resolve the inevitable conflicts which will arise regarding data needs and structures among the myriad of user organizatons.

A second problem which arises with the implicit DBA approach is that a data base is not static. As it is used, there will be a need for continuous evolution of the data base. New data requirements will arise. New relationships and structures will be needed. This evolution must be controlled and performed with the same skills as the initial data base definition. The system implementors will usually not have the time nor the charter to monitor this evoluton. But the pressure for system changes from the users will be continuous. Unless someone is explicity chartered to manage the evolution, necessary changes will not take place or will be accomplished only with great difficulty and a large impact on development schedules. This can severely impact system effectiveness and potentially even render the system ultimately useless.

Finally there will arise in the daily operation of a data base system many problems which are beyond the scope of responsibility of an EDP operations staff. These include questions of ownership of data, responsibility to populate the data base, security issues and the setting of priorities. Resolution of this type of issue requires a corporate wide, long range perspective. The operational management of a data base system, just like the system definition, must occur at an organizational level consistent with both the scope and importance of the system to the corporation as a whole.

In summary, a key factor in the successful implementation of an integrated CAD system is the highly visible, dedicated administration of the data base from the very beginning of the effort. Unfortunately this is an aspect of systems design which is all too often overlooked by technologically oriented CAD developers. But unless an integrated CAD system is effectively administered it will surely fall apart because of its own weight.

4) Don't overcontrol.

One of the primary reasons for implementing an integrated CAD system is for a company to get control of its engineering data. The integrated data base allows the company to have access to all the data, to control access to that data and to track and control the revisions to that data as a design evolves. The risk is that in attempting to gain control of its engineering data and process, the implementors of an integrated CAD system can easily go too far and make the system too rigid and therefore ineffective.

One potential control problem centers around the freedom the user of the system is given. A balance must be struck. On one hand the system's users must be constrained to adhere to corporate or department standards. This type of control is probably already in place, but the effect of it will seem much more dramatic when computer based systems are introduced, since such systems generally demand data in more precise and standardized forms than engineering personnel are accustomed to give. On the other hand, the users must be given enough freedom to be able to perform their assigned tasks in as creative a fashion as possible. The users should feel that they control the system, that it is an aid to them, not that they are controlled by it. The system designers must take care that the processes embodied in the data base and the applications built around it don't become so rigid or stylized that they restrict engineering creativity.

Another potential problem is controlling data at too low a level. For example, access and revision controls could be applied to data for an entire project, to data for each sub-assembly or to each data item in the entire design. Experience has shown that it is better to control to no lower than the scope of responsibility of a single individual. In the IDEA

System, a fundamental decision was made to force the logical design of printed circuit boards to be always in agreement with the physical design and to lock these two together in the data base. However the printed circuit layout designers are often called upon to make changes to the logic design. The decision to control logic and physical design together forced the layout designers to make all logic changes by putting the layout data back into the data base, editing the logic and re-extracting the physical data. This becomes a very time consuming process, even for trivial changes. A much better approach might have been to allow logic changes during layout and then give tools to the designer for finding discrepencies between the logical and physical implementations after the fact. Within the scope of his responsibility, the designer would then have the freedom to do his entire job efficiently. It should also be noted that system design decisions of this sort are very fundamental, they tend to propagate into many of the system utilities and applications, and are therefore very difficult and expensive to correct after the initial implementation.

5) Recognize differences between step efficiencies and process efficiencies.

Many of the companies contemplating the imlementation of an IPAD system will already be using some CAD tools. In general these CAD tools will have been built to automate some particular step in the engineering process with a concomitant improvement in time and cost for that particular step. For example, turnkey Computer Aided Drafting systems have become quite popular in the manufacturing industries becuse they can be shown to improve drawing costs and times varying from 2:1 to 10:1 or better depending on the types of drawings being produced.

A first level of integration can be achieved by tying together a set of individual CAD tools with transfer files of data. This in turn provides some improvement in the overall process by eliminating some manual steps. However the IPAD approach of using an integrated design data base goes well beyond the transfer file approach. It has been estimated that an integrated Computer Aided Design system, such as one based on IPAD has the potential to provide savings of up to 30% of the overall engineering effort. While 30% may seem a much smaller number than 1000%, it should be kept in mind that 30% of the cost of an entire new airplane engineering project is a very substantial figure, as opposed to 1000% of the cost of creating a particular type of drawing

The risk is that in order to achieve the large process improvements of an integrated system, it sometimes becomes necessary to de-optimize or add overhead to certain individual process steps. For example, if a stand-alone CAD program is already in use, integration of that program will cause a new step, data base extract and update, to be added to the user's tasks. Also control mechanisms, such as replacing a generic parts library with a standard parts library in an application, may add to the information a user must supply and limit his freedom and flexibility somewhat. It is imperative that both system developers and users understand very early in

the system design the trade-offs made to gain process efficiencies at the cost of process step overheads. If the user community does not understand and agree to such trade-offs, system acceptance will inevitably be much more difficult to obtain.

6) Top management commitment is required.

The impetus for most Computer Aided Design efforts generally comes from the lower levels of engineering management. That is where the individual design problems must be solved. But an integrated CAD system can be successfully implemented if the highest levels of engineering management and beyond are not only aware of the system, but committed to it. There are at least three reasons for this. First, an integrated CAD system will be an expensive project and will inevitably require top management sign-off. Second, the resources needed and the people affected by such a system will span all of the engineering organization. Generally the drive to obtain those resources will have to come from the top. Third, the overall engineering process and major functions will be changed by an integrated CAD system. Compromises will have to be made between process efficiency improvements and process step de-optimization. Organizational charters will change and some groups may have to be reorganized. Changes at this level can only come from top management, who understands the entire system and cannot be driven by CAD developers.

One word of warning: don't oversell the system. It is all too easy for a group of CAD developers to allow their enthusiasm for their project give top management an unrealistic set of expectatons. The vision of the pot of gold may make it easier to obtain the necesary resources to get the project started. But in the long run it is far easier to face (and help) a user disappointed with the performance of his application than it is to face a vice-president disappointed in the performance of his system.

7) Don't forget the operational requirements.

For an integrated CAD system to be effective and worthwhile, it must not only be well designed and well implemented, but also well run. This implies a great deal of planning and effort both before and after the system is first released.

As the system is being built, appropriate hardware will have to be procured and facilities prepared. Operating procedures for that hardware will have to be developed. The unique aspects of an IPAD system - a distributed network of dissimilar processors, heavy use of interactive graphics, very large distributed data bases, a high proportion of compute-bound tasks - mean that standard EDP operations procedures will not apply. Is the network sufficiently redundant? Is tape drive performance sufficient to provide daily backup? How do users job priorities get assigned?. The answers to these types of questions will be different for an integrated CAD system than for a normal business-type data processing system. But CAD people are typically not used to asking such questions and

EDP people may not realize the uniqueness of this type of application system. The system managers will have to develop a new set of backup and recovery plans, contingency plans, maintenence plans, performance analysis procedures, etc., to assure that the system runs smoothly with adequate response time and a minimum of downtime and lost data.

Another important part of day-to-day operation of an integrated CAD system is user training. Since both the individual CAD tools and the overall system will tend to be complex, it is very important that the users be well trained to take maximum advantage of the power they have available. In the IDEA System, a training plan was derived about midway through the development cycle. High level users were brought in to learn the system and then develop the courses to be given to the rest of the user community. Courses included both system overviews as well as in-depth training on individual applications. When the system was first released there were trained users ready to go. Training courses have continued on a regular basis since then to train both new personnel and those at sites just starting to use the system.

The use of design and engineering personnel (as opposed to the developers) to write and teach the courses proved very effective. Not only were they more capable of translating the system capabilities into the users' 'language', but also they were able to develop techniques for using the system by putting together combinations of individual simple commands or functions to solve particular design problems and then teaching these techniques as well. The course development during system development also provided another very effective system test and design improvement feedback mechanism.

A final point to keep in mind is that a CAD system will be constantly changing because the technology with which it deals is constantly changing. System evolution will go most smoothly if the interaction between developers and users which was established during system design and development is maintained. One appropriate way of accomplishing this is setting up applications oriented users groups. Such groups can provide an excellent source of ideas for system enhancements, for setting priorities and for exchanging techniques and experiences. To make such groups even more effective, they can be given the final approval authority for all system enhancements in their particular discipline areas.

4. Summary

The interest in and pressure to develop an integrated CAD system almost always comes from a combination of CAD tool users and CAD tool developers. Far too frequently, however, such individuals are not particularly knowledgeable in the development of applications systems and the types of issues raised by the maxims we have presented are often

overlooked. If we have any single recommendations to give the potential IPAD developer, it would be to be sure to include systems professionals, even though they have no CAD or engineering expertise, on the development team.

This paper has dealt with the potential problems of an integrated CAD system. It is intended to be taken in the spirit of preventative medicine. For all of the problems, difficulties and mistakes in IDEA, there has been a tremendous benefit to Digital Equipment from the IDEA project. We would expect a similar benefit for any aerospace company implementing an IPAD system. We hope that consideration of the maxims we have presented will make that implementation a bit easier.

IPAD: AS IT RELATES TO THE UTILIZATION OF CAD/CAM IN GENERAL MOTORS

SYNOPSIS

Betsy Ancker-Johnson
Environmental Activities Staff
General Motors Corporation

GM has held a leadership position for over 20 years in the development and application of computer-aided design/computer-aided manufacturing. Design graphics and related tooling computer software development alone have received upwards of $50 million in funds, and major new activities are operating or under development today.

The return on this investment, in terms of over 600 computer graphics consoles in daily operation on product and tooling engineering applications, has been realized several times over; and there is no doubt in the minds of GM management that these computer applications are among the most successful ever undertaken. The fact remains, however, that GM's needs are only partially met by internal development; and we are a major purchaser of turnkey graphics systems, approximately $10 million annually, with about half of our consoles being of this type. The seeming contradiction between a completely successful internal GM program and a voracious appetite as a consumer in the marketplace is the subject of this paper.

The pressure to utilize the productivity of computers starts, of course, with the unprecedented need to design and manufacture new automotive products on a scale never seen before. GM has stated recently that it will spend $40 billion over the next five years on its product program and facilities to retain its transportation leadership. As all of you have read in the newspapers recently, the U.S. automotive business is battling to remain viable and compete favorably with other world automotive centers.

The other world automotive centers do not have a technological lead on the U.S. industry. GM has every intention to see that this foreign competition does not become the technological leader, especially in the engineering activities. The goal of IPAD is to increase productivity through the total integration of computer-aided engineering functions, and we are very supportive of this activity. A system that will integrate and manage product design data, program management information, technical computation, and other engineering data management is essential to our success at maintaining this leadership role.

Returning to the role of CAD/CAM, the automotive business has many requirements similar to aerospace, but some major differences also. Manufacturing volume is totally different, hundreds of thousands vs. hundreds, so consequently the CAM portion of CAD/CAM is mainly directed toward automotive tooling and rarely toward the production of actual parts. Therefore, the comparison of aerospace and automotive becomes more direct when you substitute the production of automotive tools with the production of product parts in aerospace.

The CAD portion of CAD/CAM in automotive product engineering spans the original part definition, analysis and test of prototype parts, and part models. Part definition and physical models are given to manufacturing engineering in order to design and produce actual tooling. This subject is discussed later.

The success of CAD in product engineering starts with the ability to represent the geometry of parts within the computer. Since the early 60's, GM has successfully used "wire frame" models (lines with surface interpolation between the lines) to represent free form shapes which comprise the bulk of automotive body parts. Representations of the more analytic shapes found in engines and chassis components are handled partially by commercially available mechanical design systems. A large number of analytic shapes require representation as solids and GM is actively developing this capability. Ultimately, as in aerospace, our needs will not be completely satisfied until computer data bases are fluent in all forms of data. We strongly support the IPAD efforts to develop, demonstrate, and promote requisite data base capability in all commercial computer systems.

Design, analysis, and the testing of prototype parts and complete automotive vehicles have become much more complementary with the types of analysis that can be performed today. Structural analysis, in particular, which came to the automotive business largely through NASTRAN® and transplanted aerospace engineering, has been responsible for much of our savings in weight. Approximately 300 GM engineers are now intensively using NASTRAN® through GM-developed graphic aids for making finite element models, interfacing to NASTRAN®, and reviewing results.

Other design analysis techniques are not used as extensively as structural analysis which has been so basic to our current mass reduction program. The problem is that automotive product cycles are considerably tighter than those in aerospace, and time is available for a limited amount of analysis generally when an absolute need is demonstrated. In the case of structural analysis, both the need and the technical ability to process jobs quickly (graphics, NASTRAN®, etc.) arrived simultaneously so that product schedules were not greatly compromised. The awareness of need for many other forms of analysis (aerodynamics, fatigue, kinematics, statistical simulations, etc.) is growing in the automotive industry, but the current tools are usable by only a relatively few experts, and these results are often not responsive to product schedules. The IPAD program through the standardization of data forms and accessing methods will increase the utility of these analysis tools. The U.S. automotive industry needs these improvements to gain and maintain a significant lead in technology over our foreign competition.

Automotive product engineers pass part definition to manufacturing engineers through physical part models, traditional drawings, and increasingly through computer data transfer when the part originates within the computer systems. Between one-quarter to one-third of GM parts are aided by the computer in the design stage and this number grows continually. When the parts are developed through computer-aided design, models are produced through N/C machining and drawings produced through automated drafting machines. The economics of today's computers allows GM to effect a cost reduction on the part

design alone but the automated models and drawings represent 80-90% of the total savings in dollars and, just as importantly, in lead time in the manufacturing process.

The manufacturing engineering function for GM, responsible for tool design and tool production, is characterized by

1. Existing approximately half within GM and half outside
2. Possessing a large component of mechanical design in their work
3. Attracting turnkey graphics systems to automate their work
4. Being frustrated that computer-designed part data is not more readily usable.

These concerns are similar to those of an aerospace subcontractor which IPAD includes within its specification.

The immediate data transfer problems to/from subcontractors with turnkey graphics systems are being approached through the Initial Graphics Exchange Specification (IGES) through the cooperation of the National Bureau of Standards and industry. GM officially supports IGES as does IPAD, and we encourage IPAD to deal with the data exchange issue on a long-term basis.

In the panel discussion, a member of GM's Guide Division details the use of CAD/CAM from the point of view of a subcontractor to our car divisions (see paper 27 of this compilation). Guide is a supplier of all lamps and some soft facia bumper systems in the Corporation and is both a user of our Corporate-developed graphics system and purchased turnkey systems. We are confident that Guide's extensive experience as a CAD/CAM user will illustrate advantages and possible pitfalls.

In summary, GM has realized extensive benefits from CAD/CAM technology even though at this time our systems are not fully integrated. GM and the rest of the U.S. automotive industry, as well as the aerospace industrial complex, must be able to implement the basic concepts of IPAD in order to take full advantage of the expansion of CAD/CAM into every phase of our engineering business. These concepts must include

- A fully integrated data base
- A communications package for distributed systems in our own shops as well as with our suppliers
- Encouragement for the development and application of a solid modeling system to supplement the wire frame technology

Appreciating the similarities and the differences between the automotive and aerospace businesses, we at GM realize that the scope of this project takes the implementation outside of the range of a company even as large as GM. We join industry as a whole in supporting IPAD and related activities which will bring the promises of increased productivity to reality.

TURNKEY CAD/CAM SELECTION AND EVALUATION

TED MOODY
THE CESSNA AIRCRAFT COMPANY

SUMMARY

This paper deals with the preliminary determination of candidate integrated design systems, benchmark evaluations, and ultimate selection through an evaluation matrix.

INTRODUCTION

When it becomes apparent to Engineering Management that methodology changes must be made in order to stay competitive in the manufacturing business, one of the considerations in that methodology is immediately identified as computer aided interactive graphics. Early on, one of the questions that must be asked and answered in these considerations is the question involving mini-computer systems or main-frame based systems. Another question that must be asked and answered in these considerations is the question of user integrated or turnkey systems. The initial investigation generally can be done by analyzing individual company needs and expectations and comparing those needs with existing systems in place at either compatible or competitive companies. In the case of Cessna, most of this initial evaluation was done by way of telephone contacts with existing user companies. Additionally, in the case of Cessna one of the early-on requirements established was that of being able to directly convert the engineering data base to data usable by the Tooling Department for the purpose of manufacturing production tools. With the preponderance of both mini-computer and main-frame systems in existence, and proliferation of computer aided graphic systems that has taken place in recent years, it is important to identify company requirements so that they can be matched with a general generic type of interactive graphics system such as user integrated main-frame, turnkey, mini-computer and so on. In the Cessna Pawnee Division case, it became apparent when considering our requirements that the turnkey, mini-computer system should be the area on which our search and evaluation concentrated. As a result, our initial investigation turned towards mini-computer systems, however, because of rapid developments that came along this evaluation was expanded to consider specialized main-frame interactive systems at various points in the evaluation.

SYSTEM EVALUATION

Sales Literature

Sales literature is one of the initial sources of information about a particular system and offers the first opportunity to narrow the field of contenders to a manageable number. This assumes the evaluation team is already somewhat knowledgeable about the field. For the uninitiated, the plethora of brochures and technical descriptions leaves the team in the rather muddled state of liking best that system described last. However, if the applications areas are reasonably well laid out and the requirements created by those applications are kept in mind, enough information can be gleaned from the marketing efforts of the competing companies to recognize obvious deficiencies in major applications areas. For example, our requirement that a CAD/CAM system be useful both in design and manufacturing of our product placed a high premium on a three-dimensional data base and Numerical Control software. We deemed these a necessity for the layout and design function and for the generation of N/C tool paths for part tooling. Therefore systems whose sales information indicated weakness in these areas could be dropped from further consideration with an acceptable risk.

Demonstrations

Once the apparent match of system requirements and capabilities by sales literature has been carried as far as practical and a number of candidates have survived the weeding out process, the evaluation can move to the next phase. The number of candidates in this phase depends on how effective the evaluation team finds the sales literature in narrowing the field and on the organization's travel budget since this phase calls for visits to live demonstrations of the system operation. There are two types of these demonstrations, "dog and pony" shows by the vendor's marketing organization and observation of production sites in the field. The marketing efforts come in two varieties, district sales offices with demonstration systems installed and trade shows such as the CAD/CAM VII show in Detroit last fall. There are two avenues to the user sites as well, the relatively informal visit to a users installation or a possible invitation to a formal users group meeting. Each of these areas has its own advantage. They all give the opportunity to verify or negate the impressions obtained from the sales contacts. The vendor demonstrations are quite polished and flashy and will tend to gloss over shortcomings in performance and accentuate the stronger points. The user demonstrations on the other hand quickly get away from the system performance and tend to concentrate on the vendors support performance. The users group meetings can yield a great deal of exposure to the general attitude and specific complaints that exist in a particular vendor's user community. This is all valuable information to the would-be purchaser and must be reliably obtained before an intelligent procurement decision can be made.

Benchmark Evaluation

After all the demonstrations have been attended, and the information digested, an impression will be left as to how well each vendor matches the advertising claims. Any with obviously large shortfalls are cut from further consideration and some means of head to head comparison between the remaining contenders must be found. Since the demonstrations by the vendors are all defined by them and well rehearsed, there is too little in common for a meaningful comparison.

One obvious approach to obtain a comparison is for the evaluation team to design a benchmark problem that is simple enough to be performed rapidly by the vendor's operations and yet exercise the system in each of the anticipated applications. Since Cessna produces airplanes, applications in lofting aircraft surfaces, drafting predominately sheet metal components, developing N/C tool paths necessary to produce form tools, and performing NASTRAN analysis of the aircraft structure were selected as possible applications and addressed in the benchmark.

A two day period was allotted to each vendor to perform identical tests. During this period, each contender was asked to generate the defining lines for an aircraft wing and fuselage, determine the surface intersections, extract the envelope for a detail part defined by the surface contours, produce an engineering drawing of the part, generate APT source and CL file outputs of tool paths to fabricate the form die, create a finite element mesh, and produce data suitable for NASTRAN analysis. This very tall order was not completely filled by any of the vendors but did provide a more than adequate amount of data by which the technical evaluation of the various systems could be completed.

SYSTEM SELECTION

Technical Evaluation

As soon as all the benchmark examinations have been completed the evaluation team is faced with reducing a preponderance of data to a form usable for generating a procurement recommendation. On occasion, perhaps, the team will discover that one of the competitors carries an advantage in all areas. In this happy instance, a recommendation can be made almost immediately. However, in general, the strengths and weaknesses of the individual systems will not coincide but will instead, overlap. The sharp contrast of the ideal situation is blurred into a grey homogeneous mass with no apparent reason to select one vendor over any other. This however is not without advantage to the evaluator. This does cause difficulties in arriving at a correct conclusion, but there is also little risk of an incorrect conclusion. Virtually any of the vendors who reach the benchmark stage can provide a system with capabilities that will

enhance the operation of the prospective buyer. At this stage, however, not only system capabilities are important but also the support capabilities of the vendor become a prime factor in the decision to purchase that vendor's system.

Just as a system with deficient or mismatched capabilities is little more than an expensive toy, so too is one with excessive maintenance downtime or with poorly trained operations personnel or with a management whose implementation and application plans are not well prepared. Therefore, along with the technical evaluation of the system must also be evaluations of the hardware and software maintenance support and reliability, training programs and procedures for both operators and managers, and the availability of consulting services to help get the new user up the learning curve as quickly as possible.

So even though any of the benchmarked systems will offer advantages, there is probably one which will best meet the needs of the anticipated applications. Since most people are familiar and comfortable with numerical rankings, the evaluation team decided to generate for each of the competitors a numerical score which would indicate the proper choice. Or, in short, to generate an objective measurement of what was to a large extent subjective data.

The technique used was a multi-tiered evaluation matrix or tree as indicated in the figure. The final score for each system was derived as a weighted average of the score in each of three evaluation areas. As can be seen from figure 1, these areas are manufacturing applications, engineering applications and overall system management. Each area has several application categories unique to the area. The engineering categories are shown. As before, the area scores are a weighted average of the category scores for that area. This scheme holds throughout the evaluation tree. The score for each block is the weighted average of the applicable blocks in the next lower tier. Each category is made up of unique items and each item of unique elements, the lowest tier. Element scores are assigned on a scale of zero to ten based on how well the system appeared to meet the needs of that category's item for that element. A conscious effort was made to avoid ranking the systems relative to each other at all levels of the evaluation. The intent was to obtain an absolute rating for each system rather than a relative ranking. The evaluation team felt important consideration was not how one system compared relative to another, but rather how each system would perform in our operation on our applications.

As might be expected, no system obtained a perfect score. Of the four systems evaluated by this process, the high was 91 and the low was 72 out of a possible 100. This should not be interpreted to mean that there is in fact a 20% difference in capability but rather that the lower score indicates the system strength did not match our needs as closely as did the other systems. Both the high and low scores are from systems which are very popular and have a large user base.

These final scores from the technical evaluation were used by the contract negotiators as background information in the negotiations that followed. The negotiations were performed by different individuals from those who were involved in the technical evaluations and benchmark performances.

Contract Negotiation

Once the candidate systems have been identified, evaluated, benchmarked and technical evaluation completed, the contract negotiation phase begins. In most companies there are several prime considerations in the purchase of any piece of capital equipment. One of those certainly is a technical evaluation and the other the capital outlay required for the equipment. Generally, it is desirable during the technical evaluation to qualify at least two systems as having the necessary requirements for the applications intended. Once the fundamental systems are established in terms of their capabilities, a system definition in terms of the various components and peripheral equipment must be established. These obviously vary with respect to each company's present computer capability and peripheral equipment but generally a system definition must be established. In the case of Cessna, this was done with two competitive but equally acceptable vendors. One of the next considerations in the contract negotiations involves determination as to whether or not equipment should be outright purchase or lease. Consideration must be given to maintenance contract and service support. Once these are done, a formal quote is requested with a clear cut decision date established. During the evaluation of these quotes, the competition reaches a high level and many times suggestions for reconfigurations are made to optimize the capability and cost parameters. But in general, the equipment with the strongest technical capabilities, best maintenance and service organization, and total cost that is compatible with budgeted figures must be selected. There is a potential for significant cost reductions with two qualified vendors operating on a bid basis for complete interactive systems.

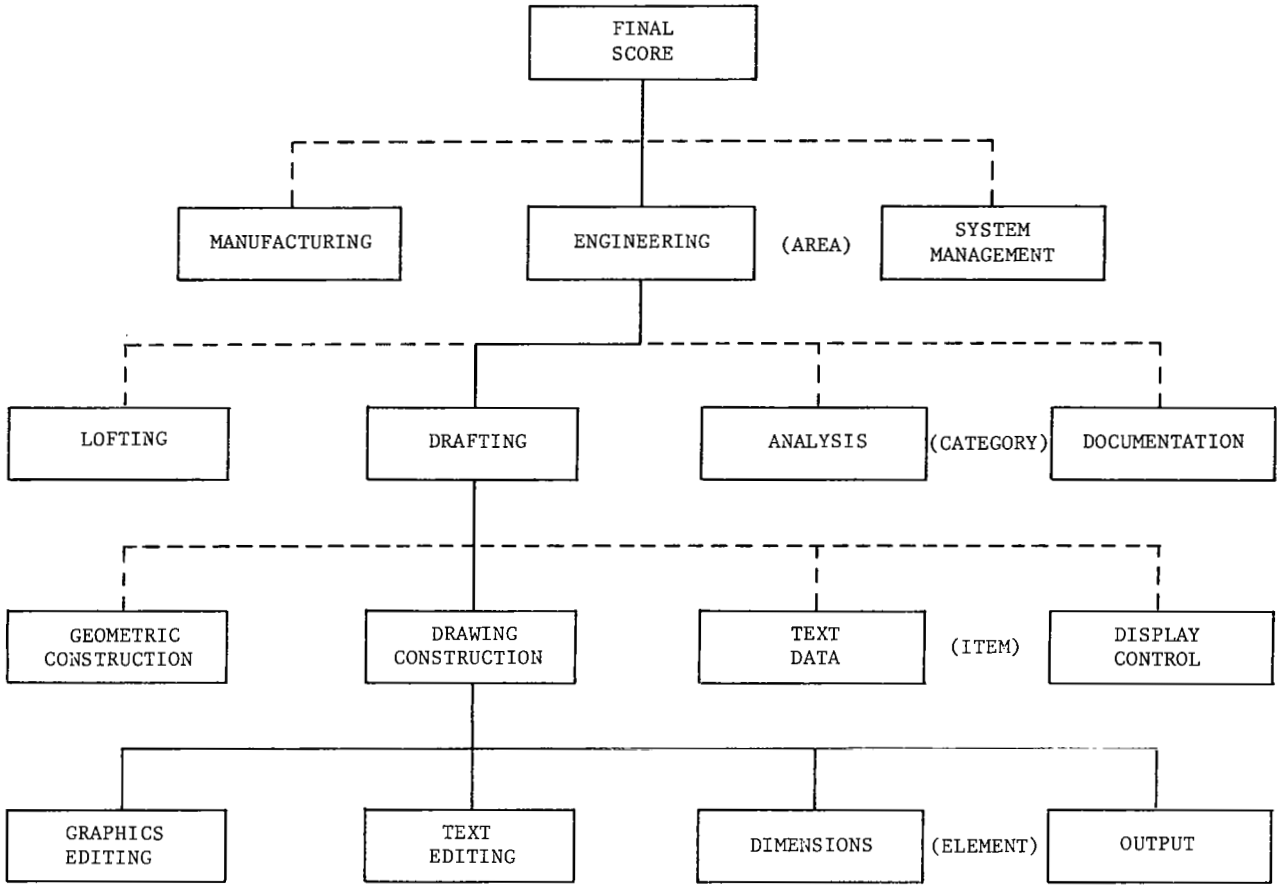


Figure 1.- Evaluation tree.

OBSERVATIONS BASED ON DEVELOPMENT OF A

COMPUTER AIDED DESIGN SYSTEM

Howard Best
Vought Corporation

INTRODUCTION
VOUGHT'S CAD EFFORT

We are all aware that use of the computer for design is not new. The technical specialties have had their NASTRAN's for years and have used them effectively in support of design. Flutter analyses which took six months on an electric calculator, are done in less than a day. So what's new that makes us take a new look at computers in the design process?

1) Graphics is reaching a state of maturity that makes it attractive for design definition.

2) Basic computer capability and capacity has attained a cost and size that makes stand above systems practical for the many designers needed to handle a project.

3) The transfer of information from one technology area to another using traditional paperwork is taking longer than the develoment of the information itself.

It is in light of this situation that Vought, like many other companies, developed a plan for implementation of a far reaching CAD program.

Our CAD program has four major thrusts:

1) Development of graphics for design definition.

2) Computerizing the transfer of information from one engineering discipline to another, "untouched by human hands".

3) Computerizing the transfer of design definition information from Engineering to Manufacturing.

4) Development of a system for management of the design process and information flow.

Some aspects of our development parallel the IPAD program. Nevertheless, a preponderance of the tasks are required regardless of whether IPAD is eventually implemented or not. Our expectation is that, if and when we implement IPAD, little or no reprogramming will be required.

The development and implementation of this program has surfaced many management related issues. As one might expect, there are more questions than answers.

Five areas will be discussed.

1) Productivity Improvements

2) Effect on Organizational Structure

3) User Training

4) Make-or-Buy Consideration

5) Transition Away from Hard Copy

PRODUCTIVITY IMPROVEMENTS

WE ARE ALL CONVINCED THAT THEY ARE REAL - ARE THEY?

Most of us have started with a single item, such as the interactive graphics scope, and developed benchmarks to prove savings. These savings were then extrapolated a long way, from the measured thousands of dollars to the projected millions in savings, to justify full systems hardware and software costs. Of course, we make full use of our company's 5-year development plan to show that improvements are not only real, but also apply to our planned business mix. However, in transition from "laboratory" to production status, we come face to face with certain truths; the optimism of developers and realism of the day to day functional organization user don't match. Reconciliation of actual savings to expectations is a must to maintain credibility.

Even interim developments such as ours represent millions of dollars in expenditures. Management is entitled to feedbacks that demonstrate the benefits. So far, we are successful. New benchmarks are developed regularly. We are looking at percent utilization of scopes for graphics verification. Records of the number of times used for each of the integrated activities, and comparison to predictions, validate at least part of our savings formula. As specific new improvements are implemented, new benchmark tests are conducted. In addition, we are providing management visibility through a series of demonstrations of developments as they occur, and have scheduled at least 4 demonstrations per year. These keep management abreast of progress. Providing management visibility keeps the program sold.

EFFECT ON ORGANIZATIONAL STRUCTURE

ORGANIZATIONS WILL CHANGE BY EVOLUTION, NOR REVOLUTION, BUT CHANGE THEY WILL

Our CAD program is already breaking down barriers between Engineering technologies and between Engineering and Manufacturing. Several trends are emerging. The true technical specialists remain as technology computer program developers, while the hardware design engineer utilizes the technology that is available to him in the computer to synthesize and finalize his designs. This suggests a movement of the technical specialist to the computer departments. The design oriented engineers will broaden their activities across discipline lines, i.e. aerodynamics, structural loads, thermal, stress, dynamics, and product design, all handled by a single individual using available interactive programs. The era of the helmet, goggles, and silk scarves may be returning to the design room.

Traditional design type releases will begin to include more information in line with Manufacturing Planning. This raises the question of whether we move the traditional designer to Manufacturing, or move the Manufacturing Planner to Engineering. The same can be said for Purchasing, Processing, Quality Planning, etc. These organizational trends are appearing already. No longer do we talk about interface with Manufacturing, as though there were a wall between us. It is now integration of Engineering and Manufacturing.

USER TRAINING

TRAINING IS A KEYSTONE TO CAD/CAM USEAGE AND BENEFITS GENERATION

Training must proceed in concert with CAD/CAM development and hardware acquisition. Without it, there will not be anyone prepared to "reap the benefits". We are already asking where we can get training materials. Since the developers are all at different stages of development with different systems, each will generate his own material for the time being. Fitting training in with productive operations and still reaping the benefits which sold our system in the first place is difficult. Full dedication of hardware to training, use of split and double shifts, as well as on the job training are necessary expedients to come up to speed. With the age of aerospace engineers increasing, we wonder if we can teach "old dogs" new tricks. Fortunately, our experience to date is that the "old days" are pushing the CAD developers, not the other way around. A planned program of training must parallel CAD/CAM development if we are to gain the benefits, and maintain management confidence.

What about the academic world? My concern is that we will stop teaching fundamentals in order to conform to the CAD world. Familiarity of the student with computers and CAD type systems is O.K. as long as he knows what he is doing and why.

MAKE-OR-BUY CONSIDERATIONS

COMPUTERS AND CAD/CAM SOFTWARE AND HARDWARE SYSTEMS ARE DEVELOPING AND CHANGING AT THE SPEED OF LIGHT. OPPORTUNITIES ABOUND FOR FALSE STARTS.

Examples of "Buy" items are numerous and represent hundreds of millions of dollars invested. IPAD itself is well into development. Air Force ICAM is rolling along. Air Force ICAD is now in the wings. Mini-turnkey graphics systems with software included are flooding the market. Main frame computer companies are now on board as well. Programming houses are on the market with significant software that fits many of our needs.

This raises two significant questions:

a) Why not wait and buy off-the-shelf for a fraction of the development cost?
b) If we go ahead, will we be compatible with the developing technology?

The answer to the first is intuitively obvious. Competition forces us to go ahead, and the lure of benefits in productivity make our mouth water. The answer to the second is not so obvious, and presents a challenge not only to those of us who are moving ahead in parallel with our industry counterparts, but also to the major developers, whether IPAD, ICAM, Mini turnkeys, software or main frame systems, to provide and maintain compatibility. The applier of these wares is king, and compatibility is mandatory.

In the long run, we will be integrating the home grown system into larger, more powerful CAD/CAM store bought systems. Therefore, we will be both making and buying.

TRANSITION AWAY FROM HARD COPY

THE USER MUST UNDERSTAND HIS DATA AT EVERY STEP IN THE DESIGN AND MANUFACTURING PROCESSES, REGARDLESS OF WHAT FORM IT IS IN.

Hard copy, as a goal, will disappear, or at best, be subordinate to the computerized data base. However, the questions raised by this approach are the same as in the cashless society many foresee. They're just not as personal, but perhaps they should be.

a) Can we see visually in an understandable way what our situation is during the design and manufacturing processes?

b) Will we have adequate access to the information?

c) Can we provide adequate security to data? Who can change it? How will we know it is being changed?

d) Will we know whether schedules are being met? What management controls and visibility are necessary?

e) If difficulties occur, can we adequately trace root causes?

f) Is redundancy required for safety of the investment?

The challenge is to provide the safeguards that will give us confidence to move ahead. We are struggling with these aspects now, and only know that we must come up with satisfactory answers.

CONCLUDING REMARKS

Productivity improvements are real, provided we are astute enough to recognize high payoff applications and not let our organization clog the works with marginal tasks.

The central organization of the future will be a coalescence of design and manufacturing planning, supported by Engineering Technology specialists on one side and Manufacturing and Processing specialists on the other side.

True design engineers, capable of utilizing the tools at their command, will need to be properly trained, and continue to be honed, if they are to be effective.

Heads-up decisions concerning IPAD, ICAD, ICAM, etc. implementation are a necessity if we don't want to re-plow our ground several years hence.

We will eventually overcome our addiction to hardcopy, and develop reliable digital storage of data subject to instant recall in a format suitable for use in a multitude of ways.

INTERACTIVE GRAPHICS

Dale Christensen
Naval Weapons Center
China Lake, CA

ABSTRACT

Interactive graphics is a broad subject having many different meanings to many different people. The Navy is presently exploring a variety of uses for interactive graphics. This paper explains how the Navy Laboratories are using and planning to use interactive graphics to do engineering related tasks, and how this interactive graphics technology will impact the way the Navy Laboratories conduct business.

INTERACTIVE GRAPHICS

This paper defines the different types of interactive graphics systems available and the capabilities of one type - the minicomputer turnkey engineering interactive graphics system. In addition, the Navy's requirements and needs for minicomputer turnkey engineering interactive graphics systems will be discussed as well as the role of the Navy Laboratory Interactive Graphics Program in bringing this new technology to the Navy.

DEFINITION

Interactive graphics can be best understood by examining separately the two words "interactive" and "graphics." First, interactive denotes some type of interaction between two entities. In reference to interactive graphics, these two interacting entities are the man and the machine. Each entity brings capabilities which complement the other. The machine is composed of a digital computer and some form of visual display. The computer has the capability to perform calculations with great speed and accuracy. The visual display allows these data to be shown almost instantaneously. The man takes these data, and based on his purpose, creativity, and experience, he is able to make business or technical decisions.

The second word is graphics. Large quantities of data on a display may be overwhelming and virtually meaningless. If these data are turned into simple graphical presentations, they become readily meaningful. Thus, interactive graphics is man and a machine working together to make decisions. As always, the man is the key element.

INTERACTIVE GRAPHICS SYSTEM TYPES

Interactive graphics systems are composed of hardware (equipment) and software (programs). There are basically three types of interactive graphics systems. The first type of interactive graphics system manipulates and displays business data. This includes, but is not limited to, preparing bar charts, pie charts, graphs, and other graphical business information. The second type of interactive graphics system manipulates and displays information for real-time applications. Aircraft simulators are an excellent example of this type. The third type of interactive graphics system manipulates and displays engineering data.

There are two subtypes of engineering interactive graphics systems. The first subtype utilizes a large host or mainframe digital computer referred to as a host computer engineering interactive graphics system. This type of system is very powerful and capable of solving very sophisticated engineering and design problems. It is traditionally very expensive to use, primarily because it requires the dedication of a large mainframe computer to the task. The second subtype utilizes a digital minicomputer. This is referred to as a minicomputer engineering interactive graphics system. These minicomputer based systems are also called turnkey systems. A turnkey system is one in which a manufacturer supplies the hardware and software as an integrated system. This type of system can be useful for doing such tasks as engineering documentation, electronics design, printed circuit board design and layout, and mechanical design layout and arrangement. It is much less expensive to use than the host computer based system and is significantly more productive than manual methods. The gross sales of the leading manufacturers of these systems have grown from $20M in 1973 to an estimated $400M in 1980. It is estimated that the annual sales by 1983 will be $1.5B.

ENGINEERING PROCESS

Before looking at how the minicomputer turnkey engineering interactive graphics system works, the engineering process should be examined. Presently, there are basically four engineering functions. They are design, analysis, documentation, and fabrication. Each function is dependent on the previous function and all are interrelated. Normally, the data required for each function is manually transferred between functions by the engineer.

MINICOMPUTER TURNKEY ENGINEERING INTERACTIVE GRAPHICS SYSTEMS

Minicomputer turnkey engineering interactive graphics systems utilize a digital data base to store information. Previously entered information in the digital data base can be used by the following functions of the engineering process. For example, geometric data entered during the design phase can be used in the analysis phase, the documentation phase, and the fabrication phase. The source of the geometric data is the same for each function. This eliminates duplication of effort between the engineering functions and serves to eliminate

manually introduced error. The digital data base is controlled and manipulated by the engineer at the interactive graphics work station.

CAPABILITIES

Presently, the minicomputer turnkey engineering interactive graphics systems have limited capability. In designing, they can be used effectively for 2 and 2½ dimensional mechanical design layouts and for arrangement drawings. There does exist some software to transfer geometric data to some standard stress analysis programs. These systems are also very effective in preparing mechanical and electrical military standard drawings. Also, the capability to design, document, and provide manufacturing data for printed circuit boards is available. Two and three axes numerical control (N/C) software exists to generate machine control tapes for N/C milling, drilling, and lathe machines.

Increased design, analysis, and fabrication capabilities are presently being developed. These capabilities will mature as more software is written and as more sophisticated hardware is developed.

NAVY NEEDS

Before the Navy can effectively utilize this interactive graphics technology, it must first understand the capabilities and limitations of the technology. The Navy must also be aware of how it will change the engineering process. Three areas which will be affected are the Navy-contractor interface, the efficiency of the engineering process, and engineering personnel productivity.

The use of these minicomputer turnkey engineering interactive graphics systems will provide better and faster communication and data exchange between the Navy and its contractors. Since the engineering information in these systems is in a digital format, it can be transmitted quickly over telephone lines between any two points. Most files can be transmitted in less than 15 minutes. The ability to pass engineering data and drawings quickly between the Navy and the contractor will reduce the time needed to resolve engineering problems and to approve engineering changes.

The acquisition process will also be affected. Traditionally, Navy contractors provide as deliverables of a contract engineering specifications and drawings on paper or on vellum master drawings. Using these systems, magnetic tape will be the exchange medium. The amount of space required to provide a complete documentation set will be significantly reduced when engineering data are provided by a contractor in a digital format on magnetic tape. Storage and retrieval will also be easier.

The use of the minicomputer turnkey engineering interactive graphics system will increase also the efficiency of the engineers involved in the engineering process and will reduce the cost of fabricating the Navy's systems. Two of the most time consuming portions of a manually designed part or parts are accurately

drawing the parts and manipulating and/or changing them to evaluate new design ideas. The computational power of the computer and the speed to accurately display the parts allow the engineer new flexibility in manipulating the parts and evaluating new design ideas quickly. The design of a part will improve as more design ideas are evaluated. As the design of a part improves, the number of changes in the design decreases. The relationship of design changes to system cost was investigated recently by a large aerospace company. It was determined that design change was the cost driver in the engineering process. This company estimated that the use of interactive graphics systems could reduce the number of engineering changes by approximately 15 percent, which resulted in a significant reduction in cost to design, document, and fabricate their products. The utilization of these systems will increase the efficency of the engineers who design, document, and fabricate the Navy's products and decrease the cost of the Navy's products.

The productivity increases associated with these systems are well documented. Productivity increases of 2:1 to 6:1 are not uncommon in the industrial community in mechanical and electronic design applications. The utilization of these systems in the Navy Laboratories will increase engineering personnel productivity, reducing cost and more importantly reducing the time that it takes to develop and produce an item for the Fleet.

NAVY LABORATORY INTERACTIVE GRAPHICS PROGRAM

The Naval Weapons Center, China Lake, CA, has been designated by the Director of Navy Laboratories as the lead Navy Laboratory for interactive graphics. The Naval Weapons Center has had a minicomputer turnkey engineering interactive graphics system for over $3\frac{1}{2}$ years. The system is used to do mechanical design layouts and to create electrical/electronic and mechanical drawings. The capability to design, document, and generate art work and N/C drill tapes for printed circuit boards will be available shortly.

The Interactive Graphics Program Office at the Naval Weapons Center has been tasked to justify, specify, acquire, and integrate minicomputer turnkey engineering interactive graphics systems at 7 Navy Research, Development, Test and Evaluation laboratories at 10 locations within the continental United States. This program is divided into three phases: development plan formulation, acquisition, and integration. The development plan was completed in the summer of 1978. The Navy Laboratory Interactive Graphics Study Final Report (NWC TP 6083) was published in March 1979. The Navy Laboratory community has requested 115 work stations at an estimated cost of $10M (FY 81 dollars). The cost savings over a five-year period (FY 82-87) have been estimated at $16.3M (FY 78 dollars). The acquisition phase is well under way. Delegation of procurement authority was received from the General Services Administration in January 1980. The Request for Proposal is scheduled for release this fall. The indefinite quantity contract award is scheduled for late FY 81. Delivery and integration will take place in the time frame from FY 82 to FY 85.

SUMMARY

The installation of minicomputer turnkey engineering interactive graphics systems at the Navy's Laboratories will provide the capabilities which will allow the engineer to more quickly resolve engineering problems and approve engineering changes, and to increase the efficiency of the engineers who design, document, and fabricate the Navy systems. This will increase the engineering personnel productivity in the Navy and in industry. This new technology will allow the Navy and its contractors to design and produce better engineered systems more quickly at a reduced cost.

DATA BASE SYSTEMS IN ELECTRONIC DESIGN ENGINEERING

Del Williams
Information Display Division
Tektronix, Inc.

Abstract

This paper covers the concepts of an integrated design data base system as it might apply to an electronic design company. Data elements of documentation, project specifications, project tracking, firmware, software, electronic and mechanical design can be integrated and managed through a single DBMS.

Combining the attributes of a DBMS data handler with specialized systems and functional data can provide users with maximum flexibility, reduced redundancy, and increased overall systems performance. Although some system overhead is lost due to redundancy in transitory data, it is believed the combination of the two data types is advisable rather than trying to do all data handling through a single DBMS.

Introduction

The invasion of computers into the Engineering discipline has brought with it a multitude of opportunities. Using the computer to increase productivity, solving problems too complex for the human mind, performing simulations, and maintaining large data bases are just a few of these opportunities. Although not being used to the maximum limits, computers are effective in increasing productivity and solving large complex problems today. In solving many of these problems, large amounts of data are needed and collected. Also, expanding into new engineering processes creates increasing amounts of data which must be managed. As the amount of data increases, the need to correlate and manage the data in an effective manner becomes a problem. A concern over the management of engineering data has surfaced within the past ten years and will continue to grow.

The end is not obvious. However, within the next decade, we will see emerging an integrated design data base concept. This will allow engineering data to provide information to various levels of both project- and management-oriented decision makers.

In electronic design alone, it is not uncommon to find many of the following elements:

- o Schematic or circuit information
- o Documentation
- o Simulation modeling information
- o Design specifications
- o Firmware
- o Software
- o Project tracking and management information
- o Reliability information
- o Mechanical specifications
- o General management information

This information is required during the R & D segment of most electronics projects. Once a project has been productized, marketing, information sales data, profit and service costs can also be added to the informational data base. The process of managing and correlating all this data has become a monumental task.

The need to manage and correlate this information has been a big problem. Although it is possible to provide such correlation with currently existing DBMS systems, the processing efficiency and time delays have been unacceptable to resource managers. If we emphasize efficiency rather than flexibility, data bases must be maintained individually, and the ability to correlate data is lost. Redundant information would also be encouraged.

The goal then is to implement an integrated design data base system which provides the flexibility necessary to manage various segments of Functional Data while allowing maximum data correlation. It should require minimal data duplication to do this while delivering maximal processing efficiency to various engineering subsystems.

Integrated Design Data Base Systems

Elements of the integrated design data base include:

- o ECB information
- o Schematic circuit information
- o Project Documentation/manuals
- o Modeling and simulation
- o Project specifications
- o Firmware
- o Software
- o Project tracking and management information
- o Mechanical drawings
- o Reliability information

This information is required in the research and development stages of all projects. If the results of the R & D are to produce a productizable output, then various marketing, sales, service, customer profile, and other post-development information will also be required. To integrate this through a single DBMS system requires extensive indexing and massive amounts of storage. Processing would require extensive data searches and CPU resources for even the simplest tasks. Also, each data subsystem would need functional as well as project indexing.

Figures 1 and 2 illustrate the concept of the integrated design data base with specialized task information and both functional and project/product slices. Since the integration of the design data base contains the various subsystemal data elements (requiring sophisticated indexes), it was decided to purchase a standard DBMS package to perform the physical management of the data.

Data Interface

To supplement and simplify the functions of the DBMS package, a specialized data base interface system was envisioned (fig. 3). The function of this interface is to bridge the gap between function systems and the conventional DBMS package. Verification of all inquiries, retrievals, and update requests is also a function of the Data Base Interface.

Working in conjunction with the data base interface is a data dictionary, which is also maintained by the interface. This dictionary contains the contents of the integrated data base along with pertinent information for the specific data sets, such as who owns the data, the frequency of updating, specific content of the various fields, security conditions of the data, and archival processes. This information is needed to provide the desired flexibility and control to make the integrated data base more efficient. A major advantage of the interface is in establishing a common communication language for all users of the integrated data base. This also allows the flexibility of replacing the current DBMS package with future and more flexible systems while having a minimal impact on all users of the integrated data base. The only modifications necessary for such a change are in the junction between the interface and the new DBMS package. The requirement for users to learn a complete new language would be alleviated.

Functional Data System

The limitation imposed by most DBMS packages circumvents their use in normal functional systems of most electronics companies. The system overhead required for even the smallest amount of data

handling is very high and precludes general use in the normal day-to-day activity. To overcome this loss in efficiency a small functional data set is created to interface directly into the specialized systems. The specialized systems include those mentioned above such as ECB, documentation, simulation and modeling. The functional data are first generated by the specialized systems and maintained in functional form until ready for inclusion into the integrated data base. The functional data structure is designed around the needs of the specialized systems. This provides the maximum efficiency to systems and user interactions. Each specialized system is designed with three thoughts in mind:

1) The human interface
2) User flexibility
3) Minimal systems and data overhead

In short, each specialized system is designed with the user in mind. Using functional data in this manner allows rapid response time and greater systems flexibility. This allows the best of both worlds: data flexibility and systems performance.

Once the functional data is ready, it is integrated into the design data base to allow its cross-referencing and correlation with other data. The integrated data base is updated daily to keep the information as current as possible. Data in the transitional state (not finalized) is flagged as incomplete during the updating process.

It is recognized and accepted that data in the functional data sets is usually redundant with data in the integrated data base. We believe that the increased efficiency in response time warranted by the functional data more than offsets the duplication. At any one time, the amount of functional data generally makes up less than 10% of the total data stored. The uses of functional data in the documentation, ECB, software, firmware, and project management systems have increased the effectiveness of those systems tremendously over systems working directly from the integrated design data base.

A general archival system which saves all functional data on a daily basis would also be implemented. Data would be updated and condensed weekly and monthly and eventually consolidated into a yearly archival system. It would be possible to retrieve any functional data modified within the previous twelve months. All permanent archiving and tracking would be done from the integrated design base. This archiving is also done daily with weekly and monthly condensation.

Management Information and Special Data Base Inquiries

Once the data has been integrated into the design data base, there are two methods of retrieving information.

1) Simple requests can be made directly into the IDDB (via the DBIF). Ad hoc and special requests are handled this way. Usually as a 24 hour turnaround basis

2) Routine management information reports (programs) would extract and reformat data from the IDDB. These tasks are usually predictable and can be scheduled.

Current Systems Status

At the present time, most of the specialized systems using functional data are in full operation. New systems are being considered with the planned implementation of one or two per year over the next four to five years. Many of the specialized systems are feeding their data into the integrated design data base. We expect to have 100% conformity to this process within the next eighteen months. The specialized systems and the integrated design data base are currently operational on a DECSystem 10 and DECSystem 20. These systems are loosely coupled via a shared disk drive. All data are completely sharable and accessible from either system. The commercial DBMS package used in handling the physical storage and retrieval of the data was purchased from National Information Systems, Inc. (NIS). The package name is DPL®.

During the evaluation phase of the concept described, we were able to determine that the efficiency and usability of the function and integrated data systems was about four times that of going with the DBMS concept alone. Also, the flexibility which allows processing ad hoc requests on an overnight basis has proved to be extremely valuable. We believe the concept, as described in this paper, gives us the best alternative in providing systems flexibility and efficiency while imposing minimal systems overhead. The largest single problem is in obtaining a "buy-in" from the authors and owners of each specialized system in allowing their data to be integrated into the central data base. Along with this integration comes certain rules which govern accuracy, integrity, and updating procedures.

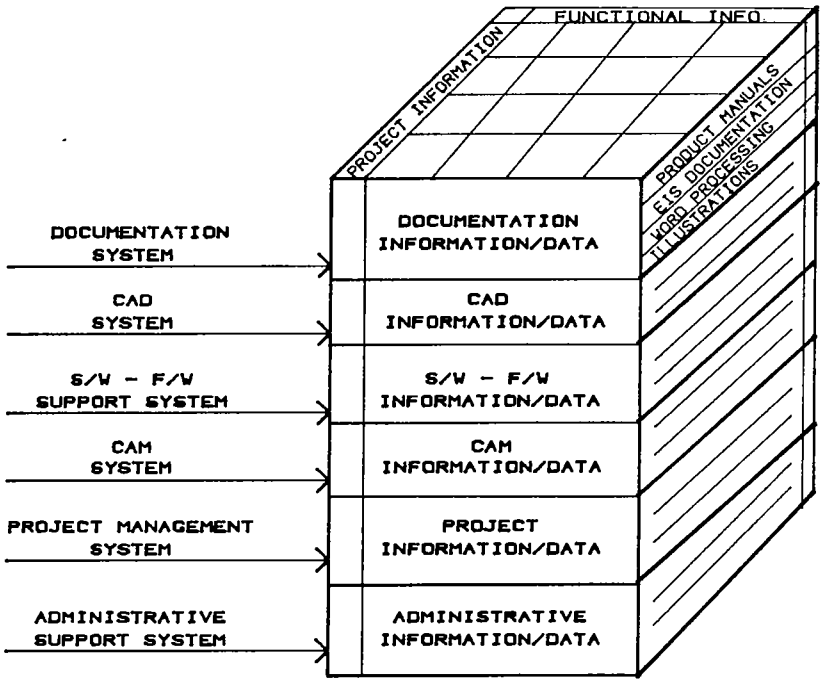


Figure 1.- Integrated design data base.

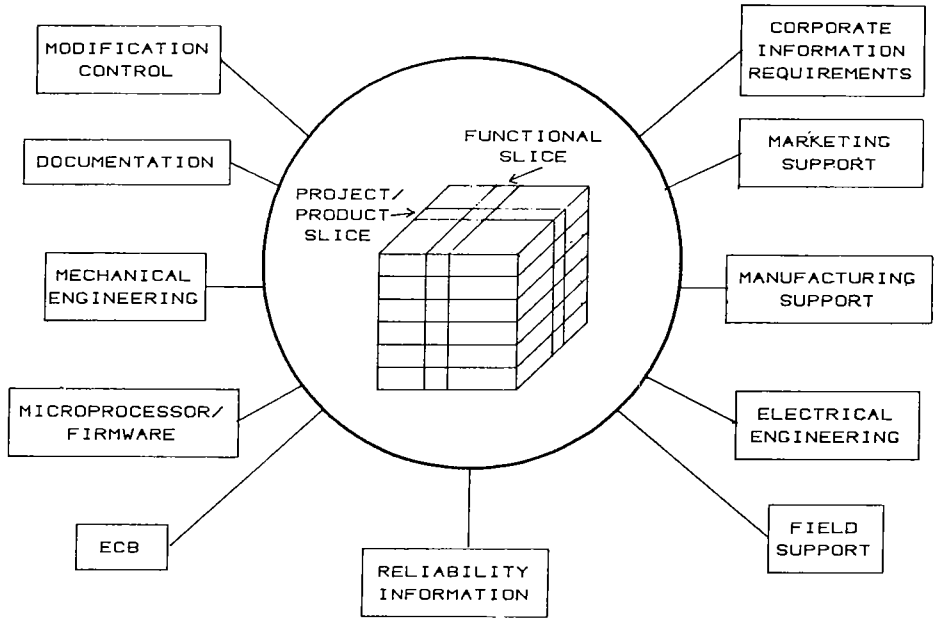


Figure 2.- Integrated design data base.

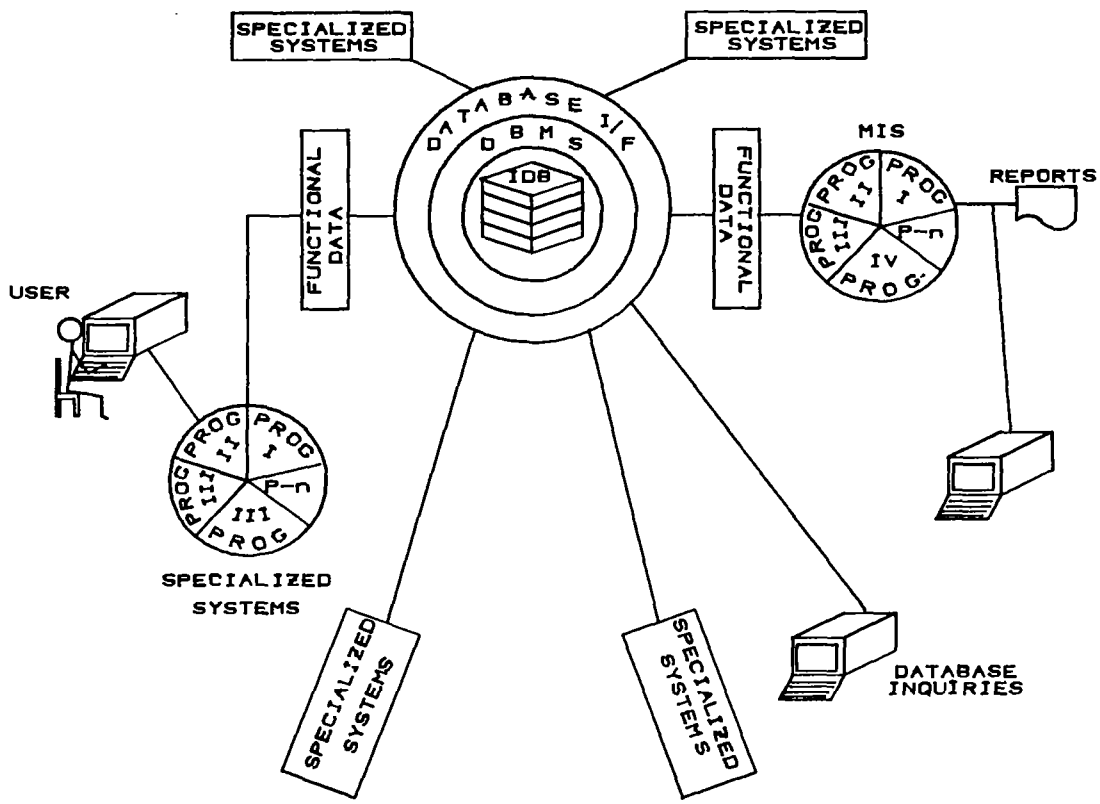


Figure 3.- Systems configuration.

SYSTEMS ARCHITECTURE FOR DISTRIBUTED APPLICATIONS

Richard Peebles

Digital Equipment Corporation
Corporate Research Group
Maynard, Massachusetts 01754

SUMMARY

This paper describes the kernel of a distributed operating system called ADAPT. The system runs on top of existing single host operating systems that are networked together. It's purpose is to transform this network of individual systems into a single system that will be easier for application programmers to use. This single system need not be monolithic; ADAPT facilitates the construction of both integrated and modular distributed systems. Note that the ADAPT system is a research project; DEC has no plans to make this a product.

1.0 INTRODUCTION

Currently a large engineering enterprise must use Computer Aided Design tools effectively in order to survive. The complexity of the design problems far exceeds the scope of the human mind. This is as true for computer vendors, who must deal with VLSI, as it is for aerospace engineers working on giant transports and missile systems. The size of these problems further dictates that the human teams working on them will be large and will represent a broad spectrum of technical skills. Most large corporations have distributed their technical staff and the result is that CAD tools to support a single large project must run on a collection of several computers networked together.

Despite the elegance of modern CAD tools (we can draw gorgeous pictures on a screen) and despite enormous strides in networking technology, the computing support environment is far from satisfactory. In fact, it borders on disaster; CAD tools have been developed as isolated pieces. To use the output of a synthetic aid in some analytic tool typically requires data conversion efforts that dwarf Hercules' cleansing of the Aegean stables. Furthermore, tools are only occasionally transportable from one type of computer to another so that there may well be five different CAD tools in use that all do the same thing.

From the viewpoint of the design engineers and project managers the situation is disastrous because they are "damned if they do and damned if they don't". The complexity of the engineering design problem demands the use of

CAD tools but the user must learn to thread them together and must know how to deal with multiple, diverse tool interfaces, with computer networks, and with multiple operating systems. All of this intellectual effort is diverted from the principal design task.

In order to simplify life for the design engineer and project manager many organizations have developed some software to help integrate the design tools. There may be a file transfer system that will reliably move data from one computer to another or there may be some form of network command language that unifies the "JCL" of the various host systems. They address only the tip of the iceberg. What is needed is an integrated system architecture, like the IPAD system, that turns the hodgepodge into a truly integrated collection of tools. We should recognize that we are still in our infancy in the space of distributed system architectures. The situation is analogous to that of data base management in the mid to late 60's; IMS and Codasyl Systems are not the whole answer. They were designed with a focus on file systems and enrich the relationships between records. It was not easy then to see the higher levels of abstraction that lead to more elegant data base management architectures. Likewise, we do not see the more elegant distributed system architectures, or if we do we cannot see how to make the transition to them. We need to examine many alternatives. The research group at DEC has considered the design of one alternative: ADAPT (Advanced Distributed Application Programming Tools). In the following sections we give a brief overview of the ADAPT approach with special focus on its kernel (IPEX equivalent).

2.0 MOTIVATION FOR ADAPT

ADAPT stands for Advanced Distributed Application Programming Tools. It is a collection of software intended to simplify the construction of distributed systems. Currently the ADAPT system is being designed and implemented by the Computer Systems Research Group at Digital Equipment Corporation.

Programming is recognized as a difficult task and the difficulties have been compounded with the development of computer networks. An application that uses resources on several machines must be structured as a network of cooperating processes. In consequence, a programmer must now think about this process structure in addition to thinking about the application logic (i.e., create processes,design the protocols, and recover from failures). Application programs are littered with statements for managing the network and process protocols that have nothing to do with the application. Debugging such process networks promises to border on impossibility unless some simple programming methodologies can be designed. This situation is analagous to the early days of data management when each programmer wrote his own I/O code and dealt with physical disk (tape) structures that had no relation to his application data structures.

In addition to the problems of constructing application process networks, it is necessary to ensure that processes in a net can synchronize their actions and that separate process networks can serialize their behavior. The latter simply means that the result of the concurrent execution of separate

process networks is equivalent to the result of executing them serially (Ref. 1). Several authors have described synchronization algorithms that can be used. Whichever algorithm is chosen, synchronization should be provided as a system level service and should not be embedded in a data base manager since coordination must be achieved over all resources, not just data.

The ADAPT system consists of four principal elements: the distributed operating system kernel, system synchronization, data management services, and application specification services. The kernel is the subject of this paper; the other three elements are mentioned only briefly. System synchronization is not in the kernel because it is an expensive service (in terms of message traffic and time delays). We wish to leave it as an option to the application programmer. Data management is such a fundamental part of any application that it must be provided as part of the operating system services. We give a brief sketch of how data management fits into the architecture in section 4. Application specification in a distributed system is largely a wide open research issue. At present the ADAPT system provides a utility called the Creator that helps programmers to set up process networks. The Creator is also discussed briefly in section 4.

Traditionally, operating systems play a dual role -- multiplexing system resources and furnishing a simpler virtual machine to the programmer. In loosely coupled networks* multiplexing is largely a static or local issue. It is not feasible to dynamically adjust the workload across processors, as the time to re-adjust the load is much longer than the time constant of the load. Our interest is in trying to design a network operating system that will provide a simpler virtual machine.

The approach taken in ADAPT and other distributed operating systems (Refs. 2, 3, 4, 5) is to assume that all applications will be structured as networks of cooperating processes whether they are distributed or not. Interprocess communication (IPC) is designed to be independent of location. That is, inter-host IPC is done in exactly the same way as intra-host IPC. Programmers deal with only one communication interface and are able to ignore the assignment of processes to hosts when they write code. As we will see in the next section, the ADAPT model also minimizes the process network aspects in the application code. Distributed applications cannot ignore the performance aspects of process location but that is much simpler to do when it is factored out of the problem of coding the application logic. The ADAPT kernel encourages this separation.

Current network architectures do part of the job. In DECnet (Ref. 6), for example, it is possible to send messages locally and remotely with exactly the same syntax. It is still the case, however, that the programmer must deal with low level process operations. At a minimum error recovery is difficult. We believe that programming can be simplified through the use of the ADAPT kernel.

* bandwidth between hosts measured in the tens or hundreds of kilobits per second

3.0 THE ADAPT KERNEL

Network operating systems may be designed to be the only operating system running in a host or as guest systems running on top of an existing host operating system (Ref. 4). Examples of guest systems are the NSW system (Ref. 7) and the IPEX system being designed by Boeing (Ref 8). The ADAPT kernel is a guest system layered on DEC's VAX/VMS (Refs. 9, 10, 11). This is described further in section 5 where the current implementation plans are sketched.

The ADAPT kernel is a direct descendent of the WEB system designed by Jim Hamilton (Refs. 12, 13). WEB is a host kernel that supports the object model (Ref. 14). The basic concept in both WEB and ADAPT is that the programmer's environment consists of a collection of typed objects each with a limited set of operators (functions) that can be applied to it.

The ADAPT kernel approximates the WEB model as a layer on top of VAX/VMS and DECnet-VAX (Ref. 6).

In its "pure" form the object model structures everything in the system as an object. Numbers, character strings, and records in files are all represented in this way. However, objects in the ADAPT system will have coarser granularity. Typically they will be things like files and data base segments. This is because we wish to exploit existing software wherever possible, a basic motivation for guest systems. If we were to set up records as ADAPT objects, for example, it would be necessary to re-implement VAX/VMS's record management system, RMS-32. We may eventually reconstruct file system software, but the existing software must be usable as well. This leads to a certain lack of aesthetic beauty; the system does not have the security properties that one would like and application structures are often more complex than they would be in a host system built using a common model. Despite these drawbacks, it is our contention that the ADAPT architecture represents a major advance in ease of application development.

3.1 The Virtual Machine

3.1.1 Objects and Types. Each object has a type that is defined by the set of operators that can be applied to it. Every object has an associated type object that supplies the definition of its operators. Specifically, the type object leads the kernel to a list of all the operators for the type. Each operator entry has a corresponding execution descriptor that provides the kernel with a specification of the process that will execute the operator. A simple case is shown in figure 1. This figure shows the object control block and associated data structures that are used in accessing an object. Object control blocks are collected into a (distributed) object control table that is used to find a referenced object. These are not part of the programmer's model, however (see section 5). The type object defines what functions must be specified and the object control block leads to the function definitions specific to a given object. This permits variant representations of a type. The code that executes the operators for a given type is called the type subsystem. The kernel does not implement these operators; it accepts requests

for operations on objects, locates the objects, and initiates the execution of the operator. The type subsystem is a user written application program.

Objects are not replicated in the system. That is, the kernel does not allow a user to create multiple copies of an object. If information is to be replicated for performance gains then it must be stored in separate objects. Replication for fault tolerance will occur but will not be externally visible as multiple object copies. A discussion of fault tolerance is a paper in itself and cannot be addressed here.

New types can be defined and instances of a given type can be created by users with authority to do so. The creation of a new type requires access to a kernel object called the root that defines the operations on type objects.

3.1.2 Addressing. Objects are referenced with capabilities consisting of an identifier and a rights vector (Refs. 15, 16). The identifier of each object is unique across all hosts and over all time. The rights vector portion of a capability specifies which of the operators on that object can be applied by the holder of the capability. Capabilities are stored in the normal address space of a program in encrypted form. The kernel decrypts them before use. Because the capability is in the user's address space he can alter it, but capabilities are over 100 bits long and the probability of producing a valid identifier or of increasing the rights on the same object is low. Given the implementation as a layer on an existing operating system there are undoubtedly easier ways to gain access to objects in the system. The intent of the capability rights vector and of encryption in this guest kernel architecture is to protect against accident more than to protect against malice.

3.1.3 Operator Invocation. A program requests an operation on an object in much the same style as a function call: f(o,p). Here, f is the operation to be executed, o is the object, and p is a parameter list. The semantics, however, more closely resembles stylized message passing (Refs. 2, 17). The invocation <f,o,p> is packaged up in an ADAPT object called an invocation segment(i-seg), the object, o, is located by the kernel, and the invocation segment is queued for the process that implements operations on that object (the type subsystem mentioned above). The calling process is given a capability for the invocation segment. When the called process dequeues the invocation it too obtains a capability for the invocation segment. The called process will execute a 'complete' function on the invocation segment when it is done. The caller can execute a 'wait' or 'test' function on the invocation segment at any point in time. A 'wait' will block further execution of the caller until the type defining system executes a corresponding 'complete'. Figure 2 illustrates the relation of the calling process, called process, invocation segment, and the execution descriptor. The data structures of figure 1 are the vehicle for locating the execution descriptor.

Invocation segments are objects like any other. Capabilities for them can be passed between processes and it need not be the original caller that executes 'wait' and 'test' operations. This permits very flexible structuring of the application process networks.

3.1.4 Encapsulation. The ADAPT kernel isolates a programmer from the implementation of an object type: the type is encapsulated. The programmer may be aware of the underlying process structure but he does not really have to deal with the complexities of process nets. Rather, he thinks of requesting function execution and being able to test completion. The implementation of an operation is hidden by the type subsystem. It is possible that an operation on an object is implemented in part by operations on other objects.

3.1.5 Primitive Types. There are certain object types that are implemented by the kernel and are needed to "bootstrap" the system. These include the following:

- o processes
- o type objects
- o invocation segments
- o execution descriptors: these are part of the type definition and specify information about the process that will execute the requested operation. Not all operators for a given type need be executed by one process. In principle, there can be one process per operator per object instance.

In practice, it will be common for there to be a separate process for each instance of a type but that process will implement all operators of the type. The latter model is the usual notion of a resource monitor or controller.

3.2 Programming with ADAPT

3.2.1 Type Subsystems. In order to define a type a programmer must create the type object, the execution descriptor(s) and the operators. This is done by invoking the 'create-type' operator on the root, passing the execution descriptor(s) and the operators as parameters. The operators are programs written in any language. (The kernel makes no attempt to deal with inconsistencies of representation in various languages. A Cobol program that passes an array parameter to a Fortran written type subsystem will likely get surprise results. Such problems are handled by ADAPT components that are external to the kernel.)

The type subsystem program obtains invocations by executing the 'get-next-call' operator on its corresponding execution descriptor. If one process implements several operators of a type then the invocation segment will need to be parsed to determine which function is requested.

There will be standard library routines available to do this. Typically, a programmer will set up his operators as multiple entry points in a subroutine; the library code will be linked in as the "main program" and will call the relevant operator after parsing an invocation segment. All operators will return to the main program at the end of processing a request. This is

illustrated in figure 3.

3.2.2 Invoking & Completing Operators. A program that wishes to invoke an operation on an object will call the kernel 'invoke' operator. This too is implemented as a library routine that knows how to deal with the "real" kernel; see figure 4. The operand capability, the operator, and the parameters are input parameters and a capability for the invocation segment. Errors in the execution of the operator and return parameters will be placed in the invocation segment itself. This information is obtained for the caller when he executes a 'get-response' function on an invocation segment that is in the complete state. It is possible to issue an 'abort' operation for a given invocation. This will succeed if the type defining system has not yet dequeued the invocation.

An additional kernel service is the 'restrict' operation that takes a capability and a function list as parameters and turns off the rights for the listed functions (if they were on) and then returns the new capability.

4.0 OTHER ADAPT ELEMENTS

As was noted in the introduction, the kernel provides a relatively primitive environment for programming; it is not a complete distributed operating system. System synchronization primitives and data management services are layered on top of the kernel. We will skip over the synchronization primitives and briefly describe how data management services can be constructed using the object model.

4.1 Data Management

Data Management services in an ADAPT system will be provided by defining data types as object types. At a primitive level we can define conventional services such as sequential files, indexed files, and the like. At a higher level a programmer can create types such as "inventory data base" with operators specific to that data structure. Then instances can be created at various sites in a network. The operators of these type subsystems contain the logic for dealing with distributed data structures such as store optimizations and search strategies.

First, suppose that the goal is to provide a distributed file system in which each file is wholly contained at a host. The objects then are the files themselves and possibly directory objects. The types will be sequential file, random file, indexed sequential file, and directory with the obvious operators. The code that implements these object types can be standard file system code with the exception of directory objects. The latter must understand that they are part of a collection of directory objects and contain search algorithms that will find the remote files. Also they must ensure that processes executing on separate hosts don't deadlock when trying to open files for exclusive use. That is, the directory or 'file-open' processes must utilize the system synchronization mechanism.

Note that there is no intention to use the object location algorithms of the kernel to find a file in the system. The kernels cannot exploit knowledge of applications to select efficient search structures. This will be typical; search strategies will often reside in data management logic, especially when the more elaborate operators of higher level data models are to be implemented. What the kernel does provide is a simpler programming model and correct operation on those rare occasions when a file object has been moved but is not recorded correctly in the directory. (Possibly because the move is temporary.) This is because the directory entries translate file names into capabilities; not host addresses.

Now if one wished to implement a distributed file system in which a single file can be spread across several hosts then the basic file operators must be re-implemented. An example of a useful file structure of this sort is a simple key file. Given a key, a read command returns (the set of) record(s) with that key value. This could easily be implemented as a multi-segment distributed object consisting of sub-objects for each segment. These would presumably be direct access files. The operators for reading and writing such a key file would have to manage the problems of concurrent access if they permitted concurrent use.

A distributed DBMS would similarly be constructed from a set of objects, each with its associated access process that implements local operations but that understands its role as part of a collection of similar objects. A distributed Codasyl set, for example, could be implemented using the index records described by Frank Germano, Jr., and representing the data locally using the local Codasyl DBMS software. The logic for implementing distributed set operations must be written by the author of the type subsystem. The DDL for such a system would have to include statements for assigning records to hosts (or the schema processor must be given assignment heuristics). Presumably the schema processor would be able to construct the required local schemata with the added index records for distributed sets. Thus the object types would be global schema, local schema, and local data base segment. The details of the operators are too lengthy to examine here.

4.2 The Creator

The Creator is an interactive utility for adding type objects to the ADAPT system. The kernel allows users to create types and object instances but this requires a detailed understanding of the structure of type objects, execution descriptors, and the structure of code for a type subsystem. Furthermore, when an object is created the kernel returns a capability for it and nothing else. It is the responsibility of the user to remember what each capability refers to. This is not terribly convenient. The Creator understands the low level structures and is designed for interactive use in creating new object types. It prompts users for information, sets up kernel data structures and maintains a type dictionary that keeps track of the external character string names for each object type and operator. It will also assist users to arrange that instances of objects are recorded in an instance dictionary (analog of a file directory). This can only be done by including appropriate logic in the code that creates instances for the type. It will be so common,

however, that we can expect library routines to be available. The type dictionary and instance dictionaries are objects and can use the kernel mechanisms in their implementation.

5.0 IMPLEMENTATION OF THE KERNEL

The kernel is implemented as an application module since it is a guest system. Depending on the underlying operating system primitives it may need some special privileges not available to normal application programs: for example, to be able to block and unblock other processes. The first implementation is on a VAX/VMS - DECnet system and is coded in portable BLISS (Ref. 18).

The kernel is responsible for moving user function invocations from the calling process to the process that will execute it. There are several possible alternatives with respect to that execution environment. It may be at the same host as the caller or it may be remote; it may be an on-going process or it may require activation. It may not be known to the caller's kernel and require a search through the object location mechanisms. In addition the kernel checks the rights of the caller to execute the invoked operation. All of this is done through data structures maintained by the kernel.

The kernel at each host is implemented as a set of eight processes with the following roles:

- The Executor is the "master" process that controls the flow of invocation through the kernel. The Executor does the rights checking, searches the object control table for the proper execution descriptor, determines when an invocation must be processed on a remote node, transmits it, and hands it off finally to the Execution Manager. In doing all of this the Executor calls on the services of the other modules listed below.

- The Invocation Segment Manager accepts invocation data from a user process, selects an available invocation segment or creates a new one, packages the data in the i-seg and returns a capability for the i-seg with the 'wait', 'test', 'get-response', and 'abort' rights on but the 'complete' right off.

- The Object Manager maintains the table of object control blocks (the object control table). When the Executor has an invocation packaged in an i-seg it next issues a 'get-copy' request to the object manager to obtain the information in the object control block, i.e., the reference to the type object, the function list, and the pointer to the underlying object.

- The Location Manager retains information concerning function invocations locally referenced that are being processed on another host. This is not strictly necessary since the

invocation segment could be located through the object control table but this is an obvious optimization (not visible outside of the kernel).

- The Namer creates new object names that are globally unique. To do this it uses the local host id. as a prefix but none of the object location mechanisms use this for finding an object. (We may later discover that trying first at the host indicated by these bits is a powerful optimization, but again that would not be known outside the kernel.)

- The Detective is responsible for finding objects that are not local and that are not known to the location manager. It contains the distributed table search algorithm.

- The Network Manager provides the mechanism for inter-host communications between kernel modules. It understands the translation from logical host name to physical machine address.

- The Execution Manager creates and interacts with type subsystem. It understands the execution descriptors, that is, how to fire up processes when necessary. It then queues invocations on the execution descriptor that is indicated. This may require unblocking an execution environment that is waiting on a 'get-next-call'. When the type subsystem issues a 'get-next-call' it receives the invocation function code, parameters, and a capability for the invocation segment with the 'complete' right on and all others off. The type subsystem also gains full access to the underlying object. If it is an ADAPT object this means a capability with all rights turned on.

These eight processes constitute the current version of the kernel. They communicate with one another using the mailbox mechanisms of VMS and the Executor orchestrates all of the activity. The principal data structures are the object control table, invocation queues, and various lists associated with processes awaiting completion notifications.

The object control table is a distributed index of known objects. Each entry is an object control block as shown in figure 1. This block points to the underlying object in one of three ways:

1. If the object is local and primitive then the pointer is a name that can be used by the type subsystem to identify the VMS representation (for example a filename).

2. If the object is remote and primitive then the pointer is a "location hint" that indicates where the underlying object was when this entry was created. (See more on this below.)

3. If the object is non-primitive then the pointer is a capability for the underlying ADAPT object.

The "location hint" in number 2, above, need not be correct. If the Detective follows the hint but does not find the object at the suggested host, it will inititate a full network search. We have not yet chosen a specific search algorithm. Rather, we plan to experiment with several. It seems appropriate to leave this algorithm somewhat flexible since the best choice will depend on network size and usage patterns. The latter are not likely predictable bult one can surely exploit size information.

6.0 CONCLUSIONS

The goal of ADAPT is to simplify the problem of writing applications in a computer network environment. Since the system is just now being built we have no experience yet to confirm or deny the value of the kernel and other tools; we can only state the ways in which we hope it will help. In the introduction we identified several problems that face a programmer who must deal with a network of operating systems. These were managing process networks, doing network I/O, location dependent forms of process interaction, and error recovery. We discuss each of these points briefly.

The application programmer who uses typed objects does not structure process networks; his view of programming is very much more like the conventional view of today. The programmer who creates a type must be aware of the process stucture since he is responsible for creating the execution descriptors. Further, if the type is itself a distributed object the interaction between processes managing the different segments of the object must be programmed by the creator of this type. Such is the case for a distributed DBMS, for example. When the segments of a distributed object are themselves ADAPT objects, even this interaction will be simplified. Finally, we note that programmers who are enamored of process networks can easily construct them using the kernel; servers can be objects.

Network I/O is hidden completely by the kernel. The kernel manages all the issues of virtual circuit establishment and of message transfer.

Process interactions are not location dependent. If a programmer uses the kernel primitives he is totally unaware of where an object is.

We have not said anything about error notification and recovery. This is a difficult topic and we cannot do justice to it here. Briefly, we must distinguish failures in the execution of an operation from a failure of an invocation. The former is reported in much the same way as a successful execution, passing error analysis information back in the invocation segment. Whatever is provided is the responsibility of the implementor of the type subsystem. Failures in invocations can occur because rights are not present in the capability, the object is unknown, the object is inaccessible, the object is known to be damaged, or the invocation takes too long. We observe that none of these need to introduce location dependencies - we can treat the errors uniformly. This is important to keeping the programmer's model simple.

The ADAPT kernel does not specifically address CAD issues. Rather, it is a base on which a whole distributed CAD environment can be built, providing

for project management and interpersonal communication as well as for CAD specific tools. It is certain, however, that neither ADAPT nor IPAD represents a final solution; they are efforts to glue together disparate pieces. Today this is the only reasonable approach; the future promises to be much more exciting if we can survive in the interim!

7.0 REFERENCES

1. Eswaran, K.P., J.N. Gray, R.A. Lorie, and I.L. Traiger; "The Notions of Consistency and Predicate Locks in a Database System;" Communications of the ACM, Vol. 19, No. 11, Nov. 1976, pp. 624-633.

2. Feldman, J.A.; "High Level Programming for Distributed Computing"; Communications of the ACM; Vol. 22, No. 6, June 1979, pp. 353-368.

3. Peebles, R.W., and E.G. Manning; "A Computer Architecture for Large Distributed Data Bases", Proceedings of the International Conference on Very Large Data Bases"; Sept. 22, 1975; pp. 405-427.

4. Watson, R.W.,and J. G. Fletcher; "An Architecture for Support of Network Operating System Services"; Proceedings of the Fourth Berkeley Conference on Distributed Data Management and Computer Networks; August 28-29, 1979; pp. 18-50.

5. Wittie, L.D.; "A Distributed Operating System for a Reconfigurable Computer Network"; Proceedings of the First International Conference on Distributed Computing Systems, Oct. 1979; pp. 669-678.

6. Digital Equipment Corporation, Decnet-VAX User's Guide, AA-D901A-TE, Aug. 1978.

7. Millstein, R.E.; "The National Software Works: a Distributed Processing System"; Proceedings ACM Annual Conference, 1977, pp. 44-52.

8. Boeing Commercial Airplane Company, "IPEX Prelimiinary Design"; IPAD Project report; August 1978, (91 pages).

9. Digital Equipment Corporation, VAX-11 Architecture Handbook, 1979.

10. Digital Equipment Corporation, VAX-11 Hardware Handbook, 1979.

11. Digital Equipment Corporation, VAX-11 Software Handbook, 1978.

12. Hamilton, J.A., Functional Specification for the WEB Kernel, Corporate Research Group internal report, Digital Equipment Corporation, ML3-2/E41, 146 Main St. Maynard, MA, 01754, November, 1978.

13. Hamilton, J.A., "Location Dependencies in Distributed Operating Systems", Proceedings of the Louisiana Computer Exposition, Lafayette, Louisiana, March 1979.

14. Jones, A.K.; "The Object Model: A Conceptual Tool for Structuring Software", in Operating Systems, an Advanced Course; Springer-Verlag; Berlin/N.Y., 1978; pp. 7-16.

15. Dennis, J.B., and E. Van Horn; "Programming Semantics for Multiprogrammed Computations;" Communications of the ACM, Vol. 9, No. 3, March 1966, pp. 143-155.

16. Fabry, R.S.; "Capability-Based Addressing"; Communications of the ACM; Vol 17, No. 7; July 1974.

17. Manning, E.G. and R.W. Peebles; "A Homogeneous Network for Data Sharing: Communications"; Computer Networks; Vol. 1, No. 4; May 1977, pp. 211-224.

18. Digital Equipment Corporation, BLISS Language Guide, AA-H275A-RK, Sept. 1978.

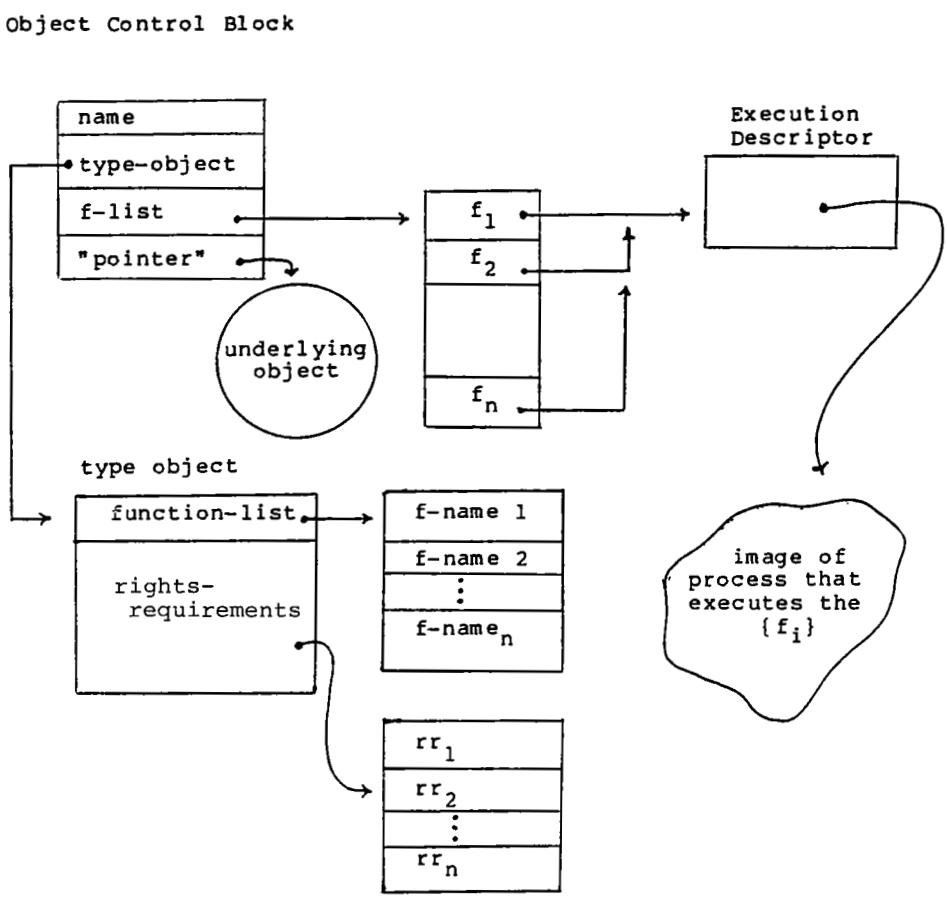


Figure 1.- Object control blocks and underlying type data structures.

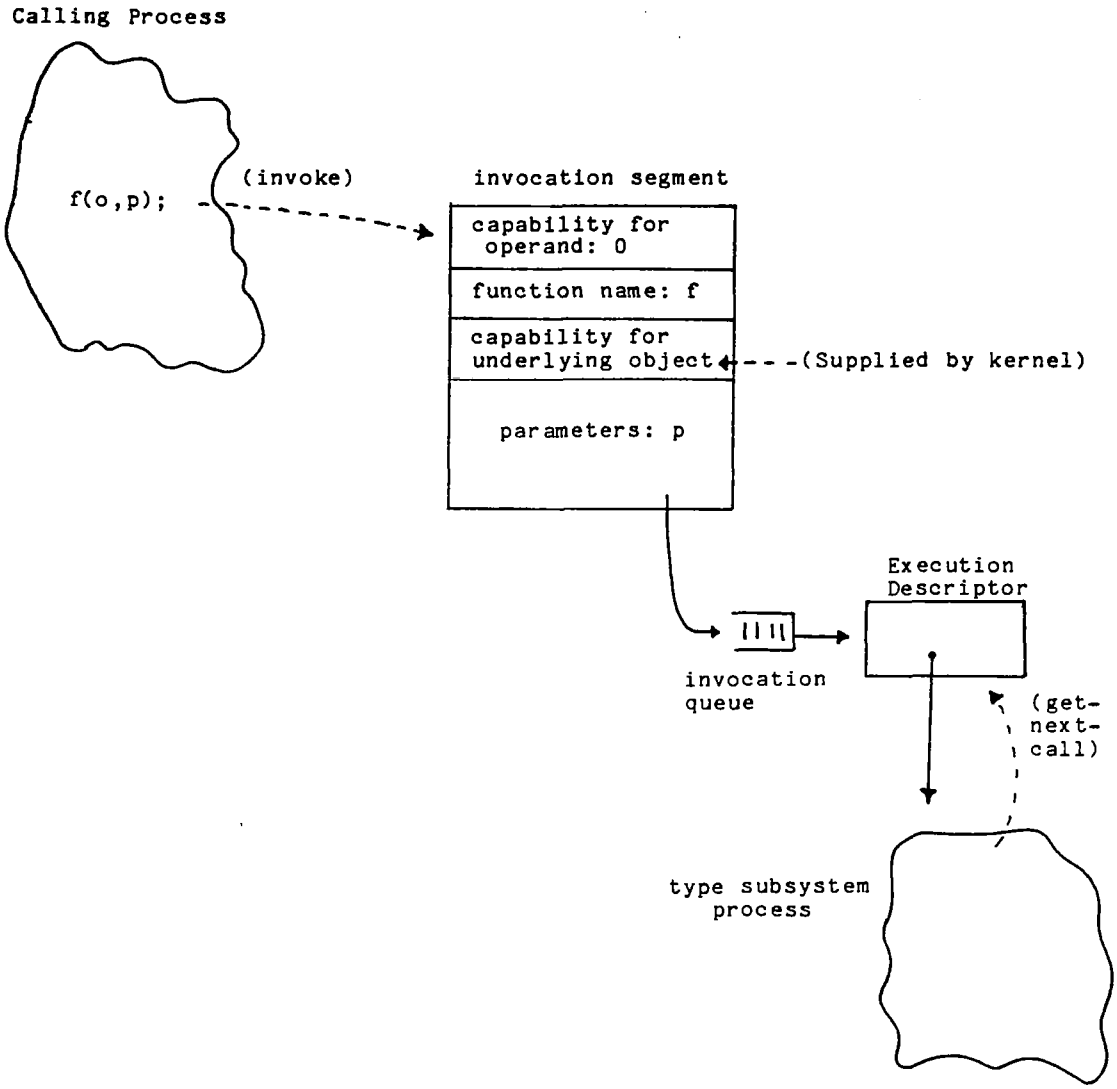


Figure 2.- Operator invocation.

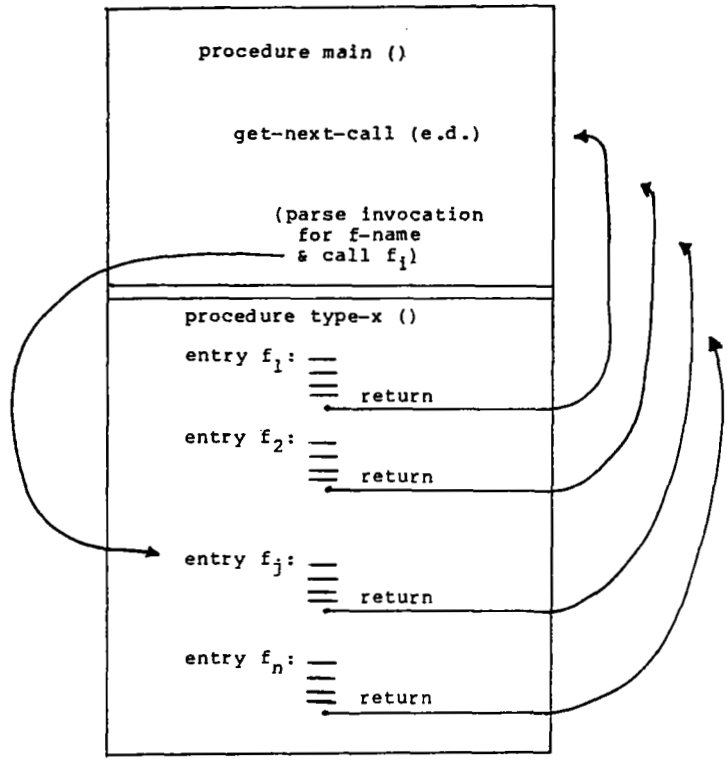


Figure 3.- A type subsystem implemented as one process.

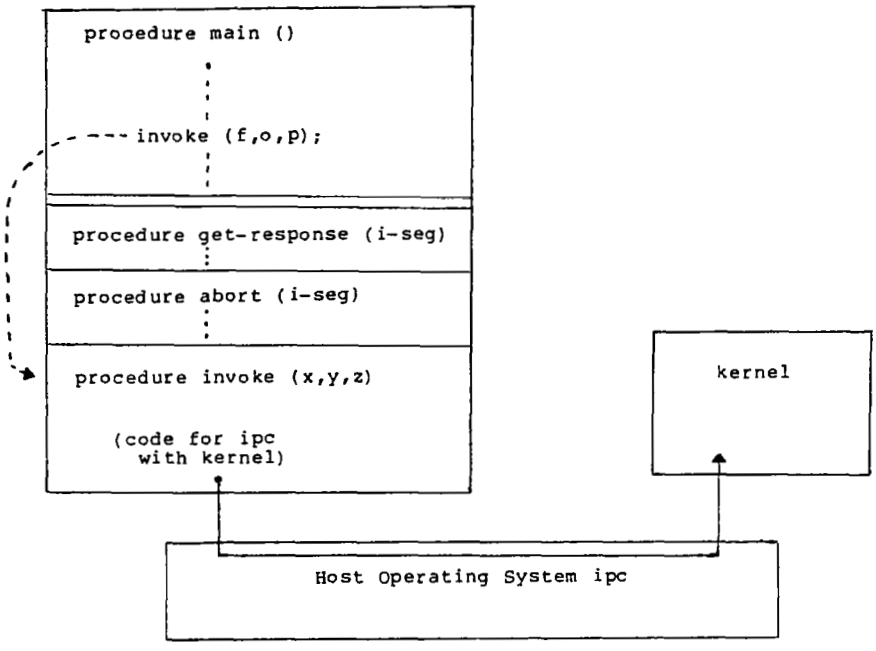


Figure 4.- Invoke library function for communication with kernel.

A REMOTE COMPUTER GRAPHICS USER

AT GENERAL MOTORS

Harry S. Murphy
Product Engineering Computer Graphics
Guide Division, General Motors Corporation
Anderson, Indiana

For Presentation at the
1980 Integrated Programs for Aerospace-Vehicle Design
National Symposium
Brown Palace Hotel - Denver, Colorado
September 17, 18, 19, 1980

ABSTRACT

Guide Division will discuss its successful use of automotive body surface design data that has been originally created elsewhere in GM's two large computer graphics systems of CADANCE and Fisher Graphics. As a supplier of exterior lighting components, radiator grilles, energy-absorbing soft-faced bumper systems, and other associated items, Guide has become most dependent on the corporate computer graphics systems to supply accurate car body styling and sheet metal surfacing information for the design of their products. The presentation includes the origin and transfer of design data to a remote user site; its use in the design of their products; and the ultimate production of detailed drawings, N/C punched tapes, and subsequent downstream transfers of detailed part data to a turnkey system for tool design purposes.

COMPUTER GRAPHICS PRESENTATION
FOR 9/18/80 IPAD NATIONAL SYMPOSIUM

This presentation will consist of three basic sections:

1) a description of Guide Division's function within the General Motors Corporation,

2) an explanation of some of the problems that have led us to the incorporation of computer graphics, and

3) how Guide utilizes graphics design data that has been created elsewhere in GM's two large computer graphics systems.

Guide Division of General Motors Corporation, headquartered in Anderson, Indiana, is the world's largest producer of automotive lighting equipment and a leading molder of plastic parts for motor vehicles. We normally employ approximately 6,500 persons at our home plants in Anderson and 1,000 more at our relatively new manufacturing facility in Monroe, Louisiana. These plants total over 324,000 m^2 (80 acres) of working floor space in producing a myriad of automotive and truck exterior lighting devices. Guide manufactures a mix of a half-million product units per day in consuming 34 metric tons (75 million pounds) of over 100 different plastic formulations in a year's time.

Although automotive lighting is a large portion of our product line, Guide also produces numerous energy absorbing soft-faced bumper systems, rear view mirrors, lighted vanity mirrors, chrome plated plastic grilles and other plated, as well as unplated special function plastic parts; Sealed Beam Headlamp Units for both domestic and foreign customers; plus automatic lighting controls such as the Twilight Sentinel and Guide-Matic, and many other items.

Whereas most of our products do affect the cosmetic appearance of a vehicle, as well as serving functional purposes, they are generally directly affected by Federal Motor Vehicle Standards with many being integrated as a part of the basic vehicle structure such as a soft-faced bumper system.

These cosmetic features are among the final items for each body style to be released from GM's Styling Studios and due to the number of new design items involved each year, it is understandable that design lead times are often very crucial. Even on a so-called "carryover car" which might be scheduled for little yearly overall appearance change, it is these particular products that normally undergo revision to permit at least some annual new car distinction. Therefore, it has not been unusual for Guide's entire product line to be revised by as much as 80% for a single model year.

Timely engineering agreement on the styling and physical aspects of each new product with GM's Design Staff (which performs the Corporation's vehicle styling function), Fisher Body Division (which furnishes the majority of the body sheet metal), and the respective car and truck divisions, is therefore quite important to enable meeting scheduled product design release dates, followed by eventual prototype, production tooling, and product delivery timetables.

With the advent of the new downsized vehicle concepts, the available real estate for the required lighting functions on the car body surface has diminished considerably.

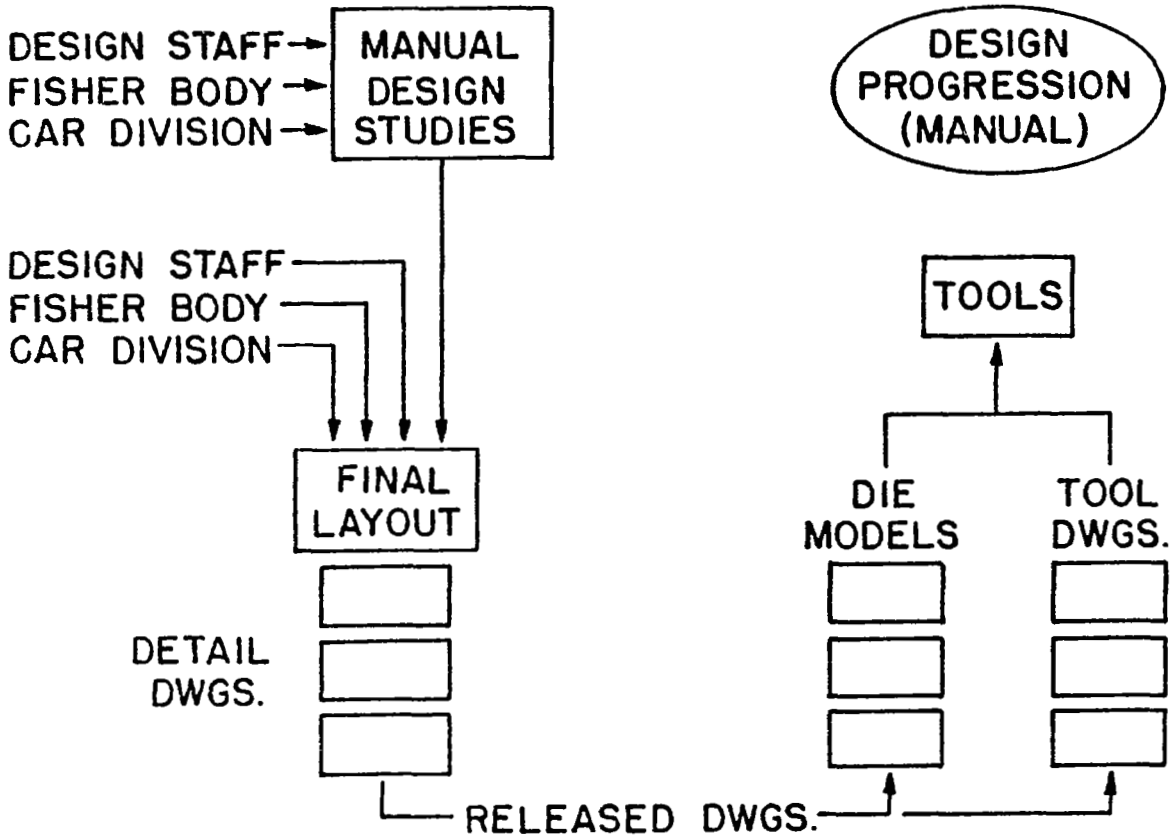
The former styling luxuries and simplified tooling facilities offered by separated lighting functions such as individual backup, sidemarker, and rear lamps are now quite often very difficult to achieve.

With the decrease in available body surface area, the vehicle stylist is being forced into many more single integrated multi-function lighting designs. The complications involved in the designing and tooling of these concepts most often result in increased design evaluation time along with creating nightmares for the toolmaker in the form of numerous slides, lifts, compound die parting angles, and other tooling complexities. Some designs require new and quite exotic fabrication and body mounting concepts as well. All of this must be accomplished with lighter materials to realize minimum vehicle weight goals and with legal and structural standards kept in mind.

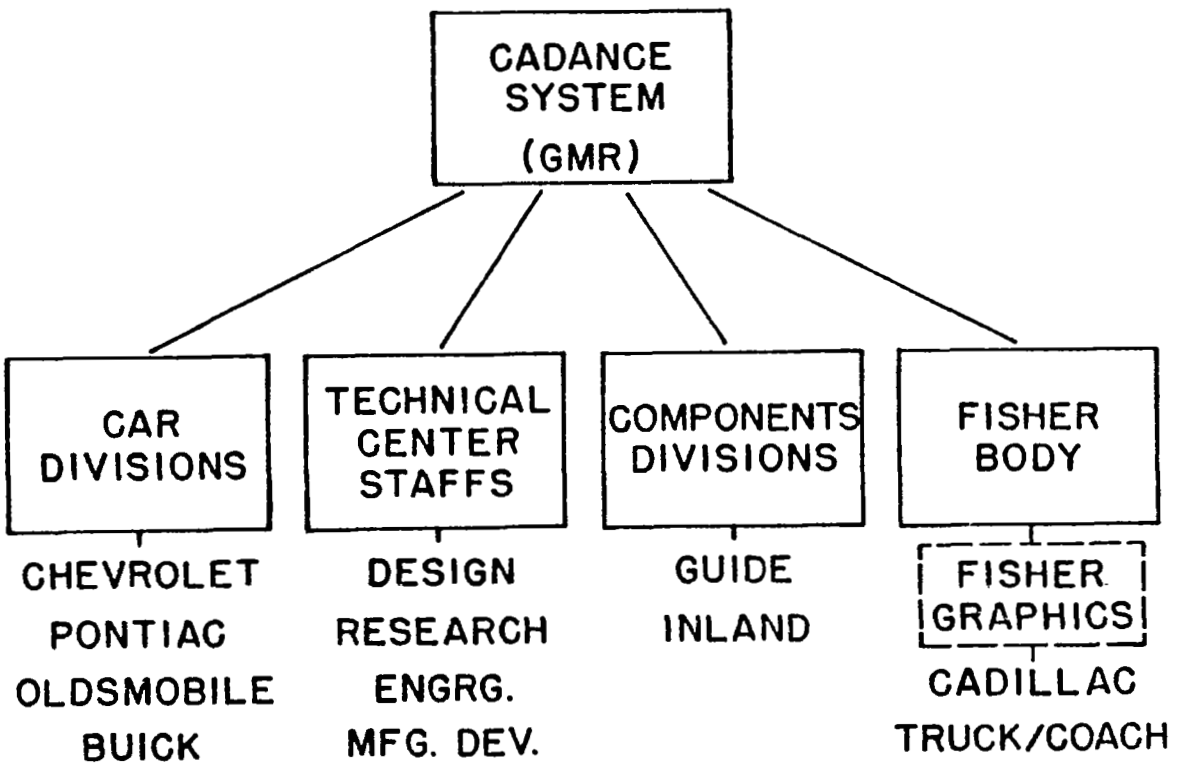
Guide utilizes as many as 120 different outside tool sources in the course of a model year to produce the tooling for our various product components. With tooling requirements becoming more complex, these sources are requesting more lead time for tool completion, which is in direct contrast to our customer's continual request for compression of design and tooling lead times.

Therefore, Guide has turned to computer graphics in an attempt to realize lead time savings in the areas of design and tooling.

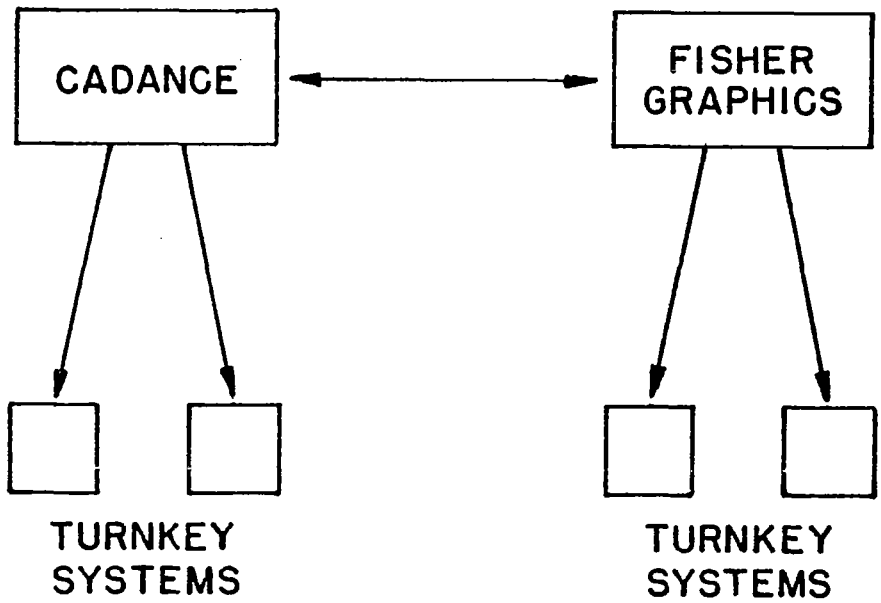
Our investigation into the capabilities of the various available graphics systems began in the spring of 1977. As the normal design progression of our products is generally quite dependent on original input from Design Staff's styling concepts plus associated sheet metal body surfacing and mounting requirements from Fisher Body and the respective car divisions, the GM CADANCE System (Computer Aided Design And Numerical Control Effort) was ultimately chosen over GM's other large computer graphics system – called Fisher Graphics — and various turnkey systems as that which best met our requirements and offered somewhat ready-made data communication links to our necessary engineering design contacts.



GM's CADANCE System is based at GM's Research Laboratories at the General Motors Technical Center in Warren, Michigan, which is located 19 km (12 miles) north of downtown Detroit. It is hosted on two IBM 3033 computers and interfaces with the various corporate divisions and staffs by either coaxial cable or telephone, depending on their respective proximities to the GM Research Labs. Over 100 graphics terminals are currently supported on the system. The largest user in the system is Design Staff which currently is operating over 40 graphics design terminals of a mix of DEC, IBM, and ADAGE devices.

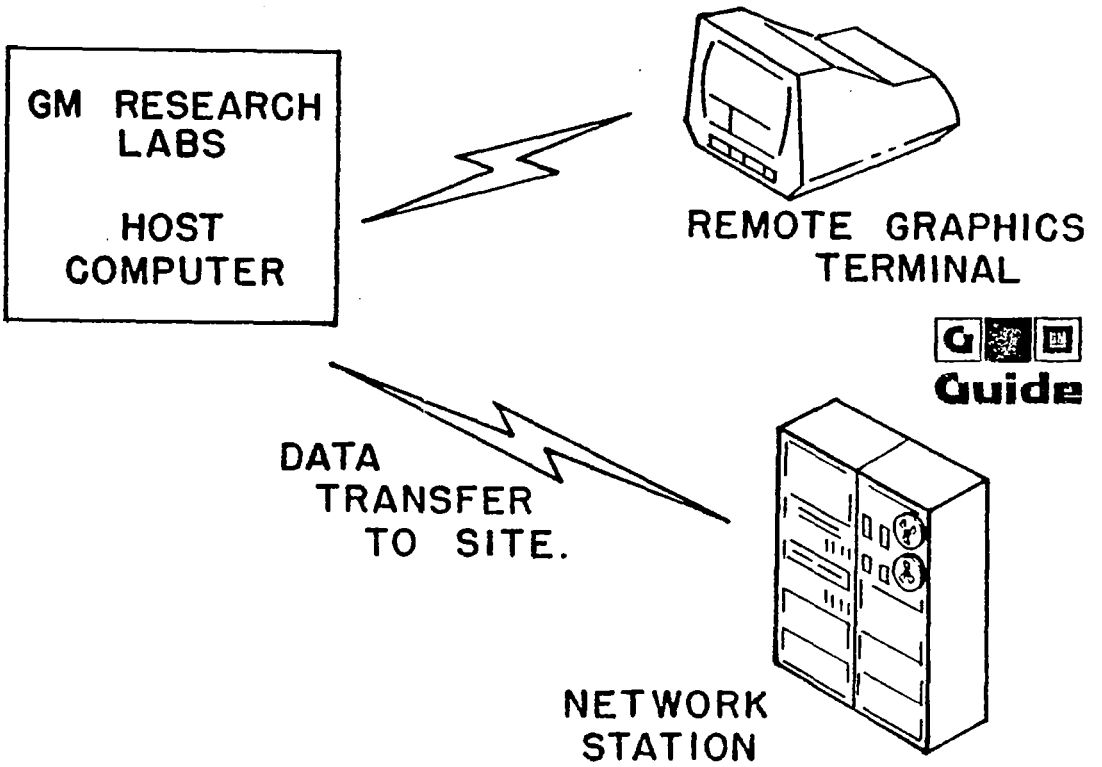


Operational interfaces between the CADANCE system and a number of commercial turnkey systems have been accomplished for sometime as well as with the Fisher Graphics System. Fisher Graphics is hosted on IBM 370's at Fisher Body Engineering, which is also located in Warren, Michigan adjacent to the GM Tech Center Complex. At present this system supports an additional 120 graphics terminals.

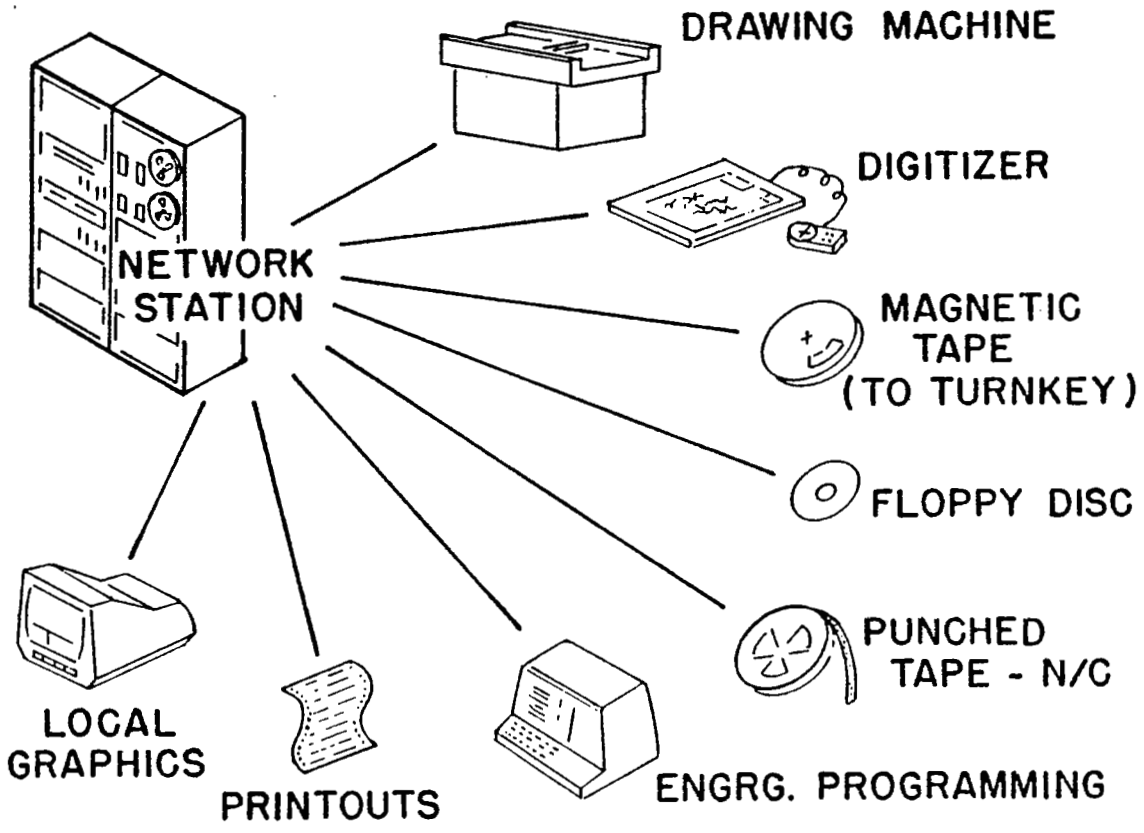


GM is now developing a common graphics system which is a merger of CADANCE and Fisher Graphics with expected completion in about two years. This single graphics system will then replace the two existing ones.

CADANCE + FISHER GRAPHICS = COMMON GRAPHICS SYSTEM



Required CADANCE produced data may be teleprocessed from the GM Research host computer to Guide or any comparably equipped outlying division via a 4800 baud phone line to a mini-computer in what GM labels as a "Network Station".



Various on-site peripherals are connected to this Network Station which can perform a variety of functions and most of them simultaneously without interruption to other Network Station user activities. This includes normal drawing production, digitizing, magnetic tape I/O for data exchange to various turnkey systems, floppy disc I/O, N/C tape punch and reader, local programming and line printer service, as well as N/C cutter tape development utilizing GM developed N/C programming on local graphics.

Guide's formal entry into computer graphics began by sending two selected designers to Manufacturing Development Staff at the GM Technical Center in November of 1977 to begin formal training and both remained until the following September of 1978 to evaluate Guide's design applications using the CADANCE system. During their stay they contributed to a dozen of our more complicated product design projects for the 1980 model year and supplied N/C die models for four major lamp components utilizing Manufacturing Development Staff's CNC milling machine.

In addition, the expertise obtained from their on-the-job activities at the Tech Center was invaluable in realizing a very smooth on-site startup with our own hardware the following December. This preliminary production atmosphere permitted the establishment of normal operating procedures, familiarity in the operation of the various necessary peripherals, and a core of reliable support personnel before the subsequent Guide facility was readied.

Four more Guide designers were sent to the Tech Center for six weeks for ormal CADANCE operator training in November of 1978. The timely completion of heir "User" training was also dove-tailed with the receipt of our initial two computer raphics design consoles and accompanying Network Station, drawing machine, and ther peripherals. This planning and the effectiveness of our subsequent hardware tartup enabled our new computer graphics facility to make an immediate contribution o our 1981 Product Design Program. Two more design consoles were added in October f 1979, #5 and #6 in June of 1980, and #7 and #8 are scheduled for an October 1980 elivery. We now train our own console operators, which currently number about 20.

When work loads demand, these qualified users operate on a two eight-hour hift arrangement which enables our six design consoles to be in constant use for 16 ours/day. This schedule permits increased productivity without the expense of overme while reducing costs per console hour of use.

The design progression of a typical Guide product begins identically whether is to be completed in computer graphics or by conventional manual methods. A esign study is manually created from preliminary drawing information received from e appropriate combination of Design Staff, Fisher Body, or the respective car divion. This design study drawing, along with ultimate design agreement and cost acceptnce from the customer, is a prerequisite to eventual hardline mylar or graphics data eleases from the required sources. The finalization of these studies often requires eeks of constant negotiation by our contact engineers with the corporate activities volved.

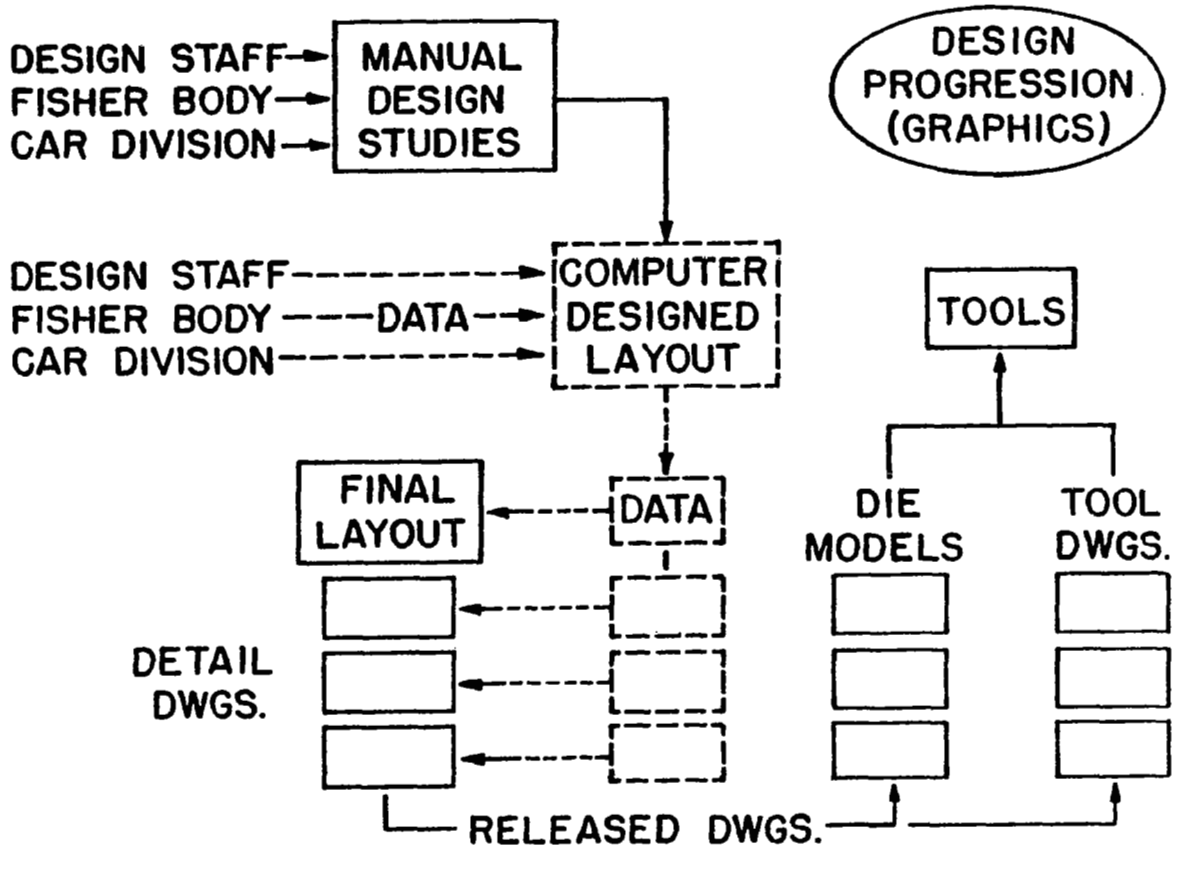
Upon finalization of the design study and receipt of the required styling and rface data, the Guide designer can immediately begin the production design of the oduct. Considerable time is saved at this point over manual methods as graphics iminates much of the tedious chore of tracing the required lines from the various ntributing sources onto a single Guide drawing before the actual design can proceed.

Upon completion of the design data for the Guide layout or final assembly, the detail of a component part can then be created in a short time by making a data copy of the entire original layout and deleting all information in that data copy except that which is required to describe the selected part. This procedure is then repeated for each individual component in the assembly.

Drawings of both the layout and subsequent details that are produced by this method are normally at least 75% complete and are then sent back to the drafting room to be manually completed. The manual tasks remaining are generally elementary and are primarily those pertaining to labels, notes, dimensioning, etc.

While these tasks are also possible to complete in the CADANCE system, our current graphics design work load has yet to permit the extra terminal time to more thoroughly complete the output drawing.

However, the current procedure has resulted in an average of 4-6 weeks savings in design time per project with up to ten weeks savings on our more complicated projects. Although our on-site graphics facility has only been active since January 1979, it will have aided in some manner in the design progression of over a third of our total 1981 model year projects and over half of our 1982 projects.



The design of products is only a single utilization of our present computer graphics installation. Four different Product Engineering areas are also using the mini-computer of our Network Station in running present non-graphics computer programs, re-writing, or writing new ones.

These consist of various forward planning, optical lens approval, material analysis, and electronic performance programs that were formerly run by our division's data processing facility. These activities are proceeding simultaneously with the required daily data transmission and drawing output design functions of our Network Station mini-computer.

Our Materials Engineering Department is also using finite element modeling features that are available within the CADANCE system in performing structural beam analysis for our Guide-flex soft-faced bumper systems. These analyses have resulted in considerable cost and time savings in the prototype phase of evaluating the effects of simulated impacts on each new bumper system.

In addition, Guide's Tool Engineering Department acquired a 4-console turnkey computer graphics system in the fall of 1979. This system is being utilized for mold design purposes and interfaces by mag tape with Product Engineering's Network Station to further utilize the part description data that has been originally developed in the CADANCE system.

The pronounced benefits of N/C related spinoffs from computer graphics developed data that are being enjoyed by our car divisions and Fisher Body in the design of car body sheet metal surfaces are not readily available to Guide at this time due to the intricate surface detail of most of our body mounted products. Although we are dedicated to develop our N/C capability to a production status, the normal surface complexity of the majority of our products has to date resulted in minimal cost/time savings and made project feasibility quite selective. Therefore, N/C is expected to remain in experimental category for yet sometime.

The utilization of computer graphics and computers for Engineering is just beginning for Guide. However, we are quite confident that the computer will play an ever-increasing role in the future of our engineering communications, our design progression and procedures, and in the ultimate tooling and manufacture of our products.

MANUFACTURING COST/DESIGN SYSTEM:

A CAD/CAM DIALOGUE

Henry H. Loshigian and Bernard I. Rachowitz
Grumman Aerospace Corporation

David Judson
Integrated Computer Aided Manufacturing (ICAM) Office
Wright-Patterson Air Force Base

SUMMARY

The development of the Manufacturing Cost/Design System (MC/DS) will provide the aerospace design engineer a tool with which to perform heretofore impractical design-manufacturing cost tradeoffs. The Air Force Integrated Computer-Aided Manufacturing (ICAM) Office has initiated the development and demonstration of an MC/DS which, when fully implemented, will integrate both design and manufacturing data bases to provide real-time visibility into the manufacturing costs associated with various design options. The first release of a computerized system will be made before the end of 1981.

BACKGROUND

In recent years the cost of manufacturing has continued to escalate in all industries. The aerospace industry has been no exception. This fact coupled with a shrinking defense budget has made cost reduction and automation to reduce cost a high priority activity. In order to achieve maximum return on investment the Air Force Materials Laboratory has embarked on a coordinated Integrated Computer-Aided Manufacturing (ICAM) Program to achieve increased productivity and thereby reduce production cost. The Manufacturing Cost/Design Guide initiated in 1975 by the Air Force provides the basic techniques and formats to allow the design engineer to perform trade studies during the early phases of a program's design activity and thereby establish an optimum design considering the cost to manufacture as a key element in the decision making process.

Figure 1 shows how important it is to conduct these trade-offs early in the design process, as it is this point in the development of a program when most of the costs are locked in, not only for production but development and operations and support as well. It is early in the program's life cycle that the economic leverage to reduce cost is the greatest.

Before proceeding too far into a discussion of the Manufacturing Cost/Design Guide (MC/DG) Computerization program, let's take a few minutes to review the ICAM program and where the MC/DG and this computerization program fit.

Figure 2 shows the "basic ICAM roadmap". Some revisions are currently being made to incorporate composite programs earlier in the schedule but the wedges shown are essentially those currently being pursued by the Air Force. The "Design Interaction" bar, shown in the first wedge, is where the MC/DG program is located. ICAM is a major effort in the Manufacturing Technology Division of the Wright Aeronautical Laboratories, covering all aspects of manufacturing including the development of Manufacturing Architectures, automated planning, simulation, language development, fabrication, automated cell/center and factory design, assembly and manufacturing interaction with the design area, etc. Current funding levels are in excess of 100 million dollars with participation of many industrial firms and universities around the country.

Figure 3 is a matrix showing many of the organizations currently involved in the Air Force ICAM program. With growing competition around the world and the need for increased productivity, Integrated Computer-Aided Manufacturing is an important area for the future of the aerospace industry.

With that very brief description of the overall program and its language, let us now look at the area of Manufacturing/Design interaction and more specifically at the Manufacturing Cost/Design Guide Computerization program. To begin with, the development and formatting of the basic costing and design data is not part of this particular program but is the subject of a separate program headed by the Battelle Laboratories under contract to the same ICAM Program Office in the Air Force Manufacturing Technology Division. Therefore this paper will not specifically address that area but will outline the computerization of the outputs of that program and the interaction of the computerized MC/DG with other ICAM manufacturing systems.

INTRODUCTION

Today's design engineer does not have tools that will provide him with quick visibility of manufacturing costs associated with his various design approach options. Too often the design engineer, under the pressure of program schedules, is forced to select a design option and release layout drawings without having the opportunity to properly evaluate the impact of his decisions on the manufacturing cost of that design.

In 1975, the Air Force initiated development of a Manufacturing Cost/Design Guide, a tool to provide the design engineer with assistance in performing design trade-offs with respect to manufacturing cost. The guide was developed by a coalition team of aerospace companies headed by the Battelle

Laboratories of Columbus, Ohio. However, its usefulness was limited by the data available and the manual effort required to use it. To remove these limitations and enhance its utilization, the development of an automated system for the MC/DG was begun. A coalition team headed by Grumman Aerospace Corporation was selected for the MC/DG Computerization program. In order to accomplish this task, the MC/DS coalition team will specifically make use of data and methodologies developed under the following ICAM Projects:

- Project 511, Manufacturing Cost/Design Guide, will provide both demonstration data and baseline requirements for the computerization of the MC/DS.

- Project 112, ICAM Architecture, will provide the current Architecture of Manufacturing as well as a standardized and approved ICAM Definition Language (IDEF) to permit the construction and analysis of MC/DS models which are such an important part of the ICAM software system life cycle development strategy.

- Project 522, Group Technology Characterization Code (GTCC), will provide the methodology for sheet metal classification required for the unique identification of manufactured items and the retrieval of information relative to these items.

The MC/DS coalition includes the responsible leaders of these ICAM Projects and their involvement is structured to optimize the transfer of data and methodologies.

PROGRAM OBJECTIVE

The MC/DG Model was created by the Air Force/Industry Coalition with the following mission requirements:

- Provide structural designers with a tool to quickly and simply obtain relative and quantitative cost comparisons of manufacturing processes

- Increase the emphasis on cost as a vital design parameter for use at all phases of the design process by emphasizing a design orientation of data presentation

- Enable more extensive structural performance/manufacturing cost trade-offs to be conducted by designers on airframe components and subassemblies

- o Emphasize potential cost advantages of emerging materials and manufacturing methods, thereby accelerating the transfer of these technologies to production hardware

- o Put the designer on the lowest cost track early in the design phase.

The primary thrust throughout these requirements is the consideration of manufacturing cost during the creative design process. This is different from the concept of estimating the cost to manufacture after a design has been completed. MC/DG is not a tool for the cost estimator; it is not a Cost-Estimating Manual (CEM). A CEM will not illustrate or emphasize cost-drivers since no design trade-offs are conducted using a CEM. Performance/ cost trade-offs must be done by the design engineer since the integrity of a performance compromise can only be evaluated by the engineer. The format of a CEM is not suited for use by design engineers, and results in time-consuming calculations which create conflicts with design schedules.

The MC/DG Model has been created and the Air Force has undertaken the task of providing a data base which will eventually include all parts that comprise the airframe as well as all manufacturing processes. It is obvious that the sheer volume of information required will become self-defeating from a variety of standpoints. Thus the need for the development of a computerized MC/DG is apparent. The concept for developing a computerized MC/DG has been evolving during current and previous MC/DG contracts and a concept validation study has recently been completed in which a prototype implementation was undertaken. An MC/DG systems analysis was not performed as part of the prototype development; rather, a series of assumptions were made based upon a limited survey of user requirements. This is in contrast to the ICAM approach of using IDEF methodology to obtain a total system model prior to the identification of a system mechanism or user, with subsequent analysis on the user level. A complete model analysis will show more readily how a computerized MC/DG might be constructed for the aerospace industry from presently available computer software components, and integrated into company CAD/CAM systems. Any long range plan for a full-scale computerized MC/DG system must allow for this type of integration within each aerospace company's facility in order to permit the future growth of the MC/DG data resource through access to other CAD/CAM data bases.

The prototype computerization has shown, without qualification, that a computer is required in order to avoid a variety of practical problems inherent in a hardcopy MC/DG. Some of the more obvious benefits of this effort are:

- o Computerized data-base updates are more practical than the publication of updated hardcopy guides. This improves the accuracy and therefore the believability of the system.

- Designer productivity is increased because he will have more rapid access to required data using representations friendly to his requirements.

- An additional benefit will be the ability to interface with other CAD tools and data used to support the designer's tasks. It is therefore required that cost-estimating information be interactively available while the designer is using his CAD system rather than having to manually refer to handbooks after a design is completed.

DEVELOPMENT PLAN

The ICAM software system life cycle development strategy represents a four-stage process in which each stage is further divided into two steps. Phase I of the MC/DG computerization corresponds to the first of these life cycle stages, with the additional requirement for an evaluation of existing software and for a conceptual design. Phase II of the MC/DG computerization encompasses the remaining three stages of software system life cycle development. Figure 4 shows the ICAM software system life cycle and compares it with the MC/DG computerization plan.

Phase I is the MC/DG Systems Analysis and Conceptual Design Development. It will define the MC/DG environment and determine the requirements for and specify a conceptual design of the computerized MC/DG system and interfaces. Phase I will result in an MC/DG System Model and a conceptual design for the computerized MC/DG. The system model not only provides a basis for the creation of the conceptual design but it also provides a view of MC/DG relative to a total ICAM architecture. The IDEF methodologies will be used to develop models of the MC/DG and the design engineer which satisfy system requirements established from surveys and previous MC/DG studies. These models are independent of any implementation solution and will also serve as the basis for understanding all of the MC/DG interfaces with other ICAM subsystems. Several implementation strategies will be developed as conceptual designs or system solutions for computerization and be evaluated for potential implementation. These system solutions will include consideration of existing public domain software to meet the MC/DG needs. The development of the computerized MC/DG is part of an overall ICAM development activity, and lines of communications will be established to assure integration of the MC/DG into the ICAM architecture. The Phase I activity will conclude with the presentation of alternative conceptual designs which best meet the requirements defined during this phase.

Upon the completion of Phase I, the Air Force may exercise an option to proceed with Phase II which is the Conceptual Design Implementation of the computerized MC/DG. This will require the MC/DS coalition to establish the preliminary and detailed designs, develop software, and test, implement and demonstrate the computerized MC/DS in the aerospace design/manufacturing environment on two host computers (IBM and CDC). It is important to emphasize at this point that the intent of the host computer demonstrations is a demonstration of the MC/DG as a corporate tool in the engineering environment and not only a validation of computer code on the mainframe products of two different computer manufacturers. This represents a far greater challenge than the creation and checkout of computer code and is the reason that the team includes five airframe manufacturers. We, the end users, have a very special interest in the success of this program since it represents a technology advance in the design process.

SYSTEM USAGE

In order to describe the program and how it fits into the overall scheme of Integrated Computer Aided Manufacturing in the short space allotted, Figure 5 was prepared as a summary. The lower portion of this figure depicts the MC/DG spanning both the Design and Manufacturing Architectures.

An example of the Manufacturing Architecture is shown in the Figure 6 node diagram "ICAM Composite View of Aerospace Manufacturing". The triangle that depicts the Manufacturing Cost/Design Guide overlaps both Architectures demonstrating that the Guide plays a definite role in both Manufacturing and Design. The computerized Guide will allow the design engineer access to manufacturing information relative to production cost for performing trade studies. Once integrated with other manufacturing systems, such data as material and machine availability, etc., will provide the design engineer with the information required to make the optimum design decision earlier, thereby reducing the need for changes. The upper portion of Figure 5 very simply shows some of the events that will take place in developing the part cost and performing the computerized trade-offs. The MC/DG System will eventually be provided with the part shape (geometry) and description from a Computer-Aided Design system. This description will be provided via the ICAM Group Technology Characterization Coding (GTCC) system which will fully describe the part. Utilizing the data developed as part of the MC/DG or company data development activities, the computerized MC/DS will display an output format for the design engineer. Depending on the independent variable (in this case part length) the lowest cost to produce will be calculated and noted. As stated previously this is a very simple example of one of the possible outputs and how the system will use other ICAM subsystems to develop a required output.

Figure 7 is a node diagram showing at a very high level the ICAM subsystem interfaces with the MC/DG program. As can be seen, the MC/DG will interface in many of the major ICAM program areas, having a major impact in the Application Systems activity. As a key element in providing manufacturing information to the design engineer relative to planning, manufacturing control, material handling, etc., the MC/DG will provide the Design/Manufacturing real time interface that is so important in reducing costs.

AEROSPACE SURVEY

A key task in structuring the overall system is in making sure that the requirements of the user and his or her supervisors are accounted for in generating a system that will be both accepted and useful. Therefore, a survey has been prepared and distributed to 57 companies, representing small, medium and large manufacturing organizations.

Figure 8 describes the composition of the survey package. Each survey administrator is given a complete package describing why and what the survey is all about, stating the purpose and viewpoint as well as questions tailored for each of three types of individuals: (1) Design Engineer, (2) Design Engineering Manager and (3) Data Processing Manager.

Figures 9 through 11 show sample question pages from each of these survey areas. A computerized survey analysis is being performed as of this writing and will be used to establish and prioritize MC/DS requirements.

IDEF MODELS

In order to insure successful development, one must first establish an integrated model of the MC/DS. Required are models that define in a very specific manner the following:

- What we are doing
- What is needed to do it
- How we do it

These three general representations have been formalized with the development of the IDEF methodologies and provide, through conventions tailored to the needs of each, a solid baseline for their construction and analysis. The existence of a Function Model ($IDEF_0$), an Information Model ($IDEF_1$) and a Dynamics Model ($IDEF_2$) provides a complete definition of any system. The Function Model provides the static representation of the system's activities and performer relationships, and the Information Model provides a static representation of the information required and its logical structure.

The primary purpose of the Dynamics Model is to capture the precedent relationships and the dynamic properties of the system. The dynamic properties are important time-oriented elements of the system. A secondary application of the Dynamics Model is to provide control properties of a system. These control properties define when an activity can begin and end, which have precedence and which can be conducted in parallel.

Figures 12 and 13 are example Composite View $IDEF_0$ drawings. Further decompositions have been made which break down these activities to very specific tasks. This model together with the corresponding $IDEF_{12}$ models under development will be used to generate MC/DS conceptual designs.

IPAD/ICAM IMPLICATIONS

The current long range NASA roadmap for IPAD shows the integration of the computerized MC/DS into an IPAD environment. This milestone will be one of the first such planned links between IPAD and ICAM. The importance of building these bridges has been formalized by ICAM through a group of roadmap projects identified as the Design-Manufacturing Interactions Thrust. It is within this thrust area that the MC/DG efforts are being funded. Joint AF/NASA efforts are now underway to identify and evaluate the criticality of key IPAD/ICAM interaction points.

The computerized MC/DS must accomplish interfaces with CAD geometric modeling systems as well as with manufacturing information systems. Fortunately MC/DS will not bear the burden of this requirement alone since the success of many CAD/CAM systems also depends upon this multiple interface. The result of the initial IPAD integration of the computerized MC/DS will be to further demonstrate the need to create standards/conventions for geometric modeling and manufacturing information management. Once such conventions are established within a company/industry, the MC/DS can function as a real time design-manufacturing cost optimization tool.

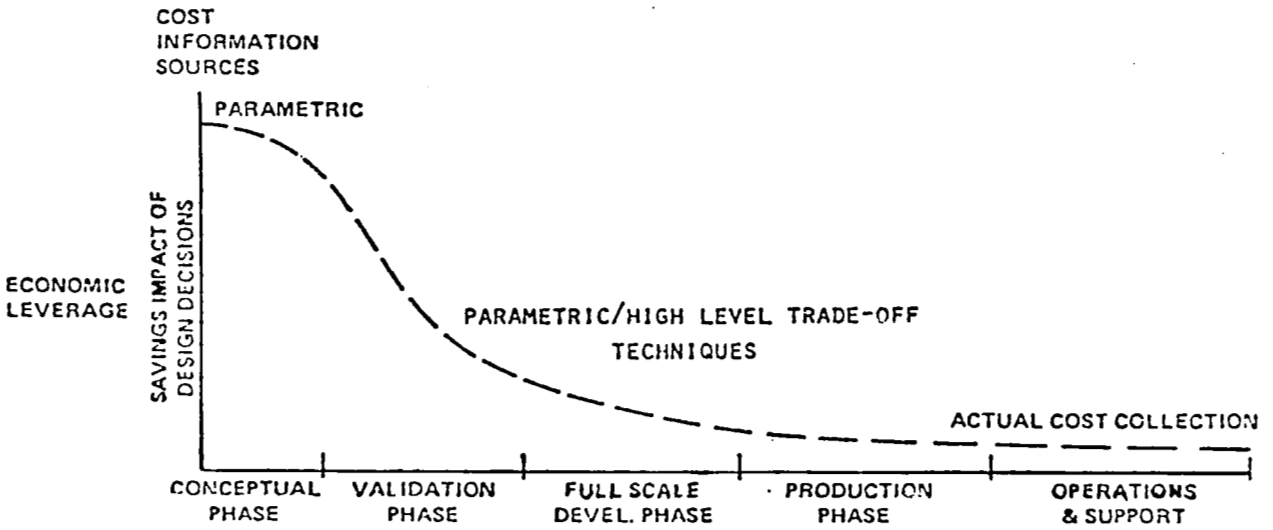


Figure 1.- Early design - economic leverage.

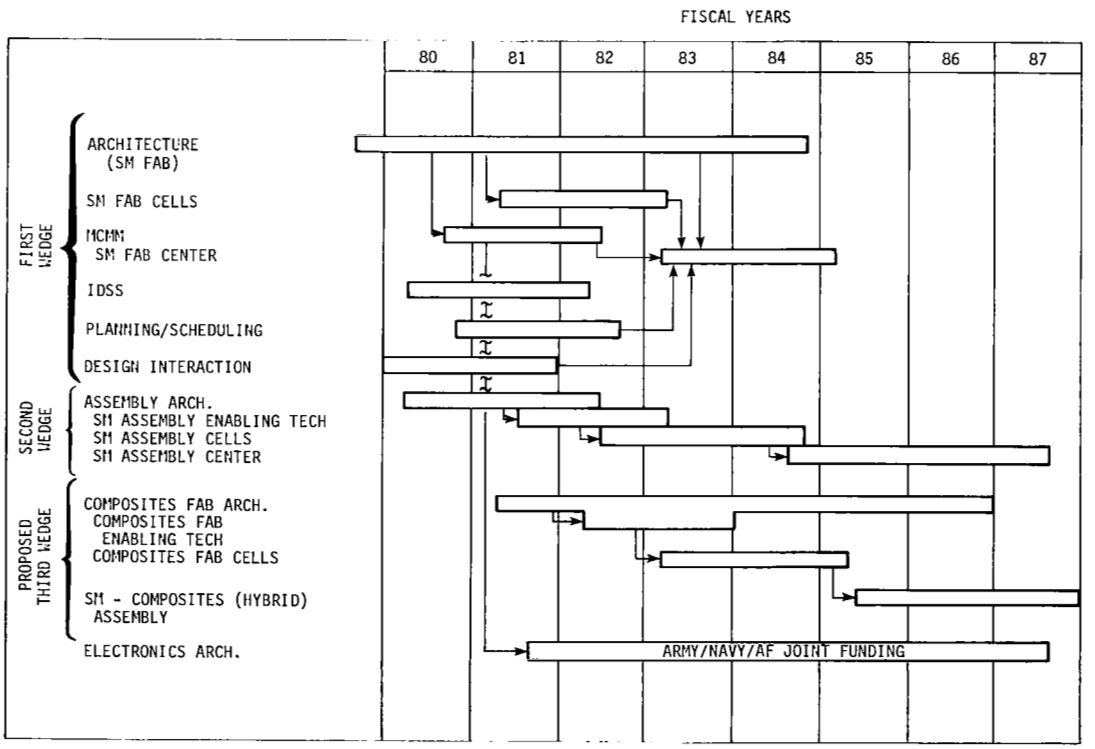


Figure 2.- Basic ICAM roadmap.

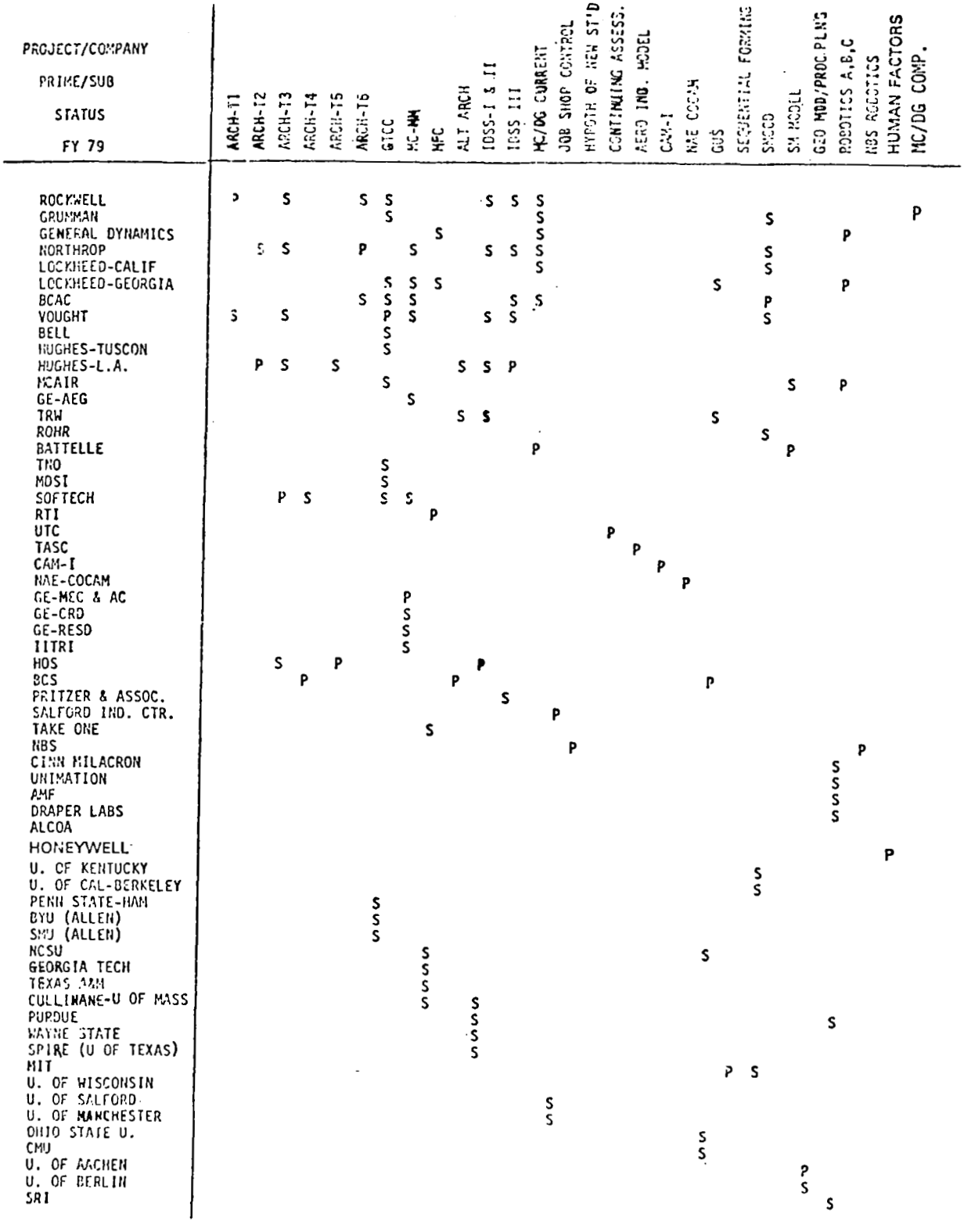


Figure 3.- ICAM matrix.

ICAM OFFICE GENERALIZED SOFTWARE SYSTEM LIFE-CYCLE DEVELOPMENT STRATEGY	ICAM OFFICE MC/DG COMPUTERIZATION PLAN
FIRST PHASE - UNDERSTANDING THE PROBLEM	PHASE I - REQUIREMENTS
STEP (1) - NEEDS ANALYSIS	(1) INTERFACES & INTEGRATION REQUIREMENTS
STEP (2) - REQUIREMENT DEFINITION	(2) DEVELOPMENT & ANALYSIS OF SYSTEMS MODELS
	(3) AEROSPACE SURVEY RESULTS
	(4) MC/DG SYSTEM SOLUTION
	(5) DATA ADMINISTRATION SOLUTION
	(6) EVALUATION OF EXISTING SOFTWARE
	(7) CONCEPTUAL DESIGN RECOMMENDATION(S)
SECOND PHASE - FORMULATE & JUSTIFY THE SOLUTION	PHASE II - OPTION 1, CONCEPTUAL DESIGN IMPLEMENTATION
STEP (3) - PRELIMINARY DESIGN	(1) PRELIMINARY DESIGN & APPROVAL
STEP (4) - DETAILED SPECIFICATION & OBTAIN APPROVAL TO BUILD	(2) DETAILED DESIGN & APPROVAL TO BUILD
THIRD PHASE - SOLUTION CONSTRUCTION	
STEP (5) - CONSTRUCTION, VERIFICATION TEST	(3) DETAILED DESIGN IMPLEMENTATION
STEP (6) - INTEGRATION, VALIDATION	(4) VERIFICATION TEST AND VALIDATION
FOURTH PHASE - INSTALLATION AND USE	
STEP (7) - IMPLEMENTATION, USER ACCEPTANCE	(5) IMPLEMENTATION AND USER ACCEPTANCE
STEP (8) - MAINTAIN AND SUPPORT	(6) TRAINING MECHANISMS DEVELOPMENT
923-31	(7) EVALUATION OF "PLATO" FOR TRAINING

Figure 4.- ICAM generalized software MC/DG computerization comparison.

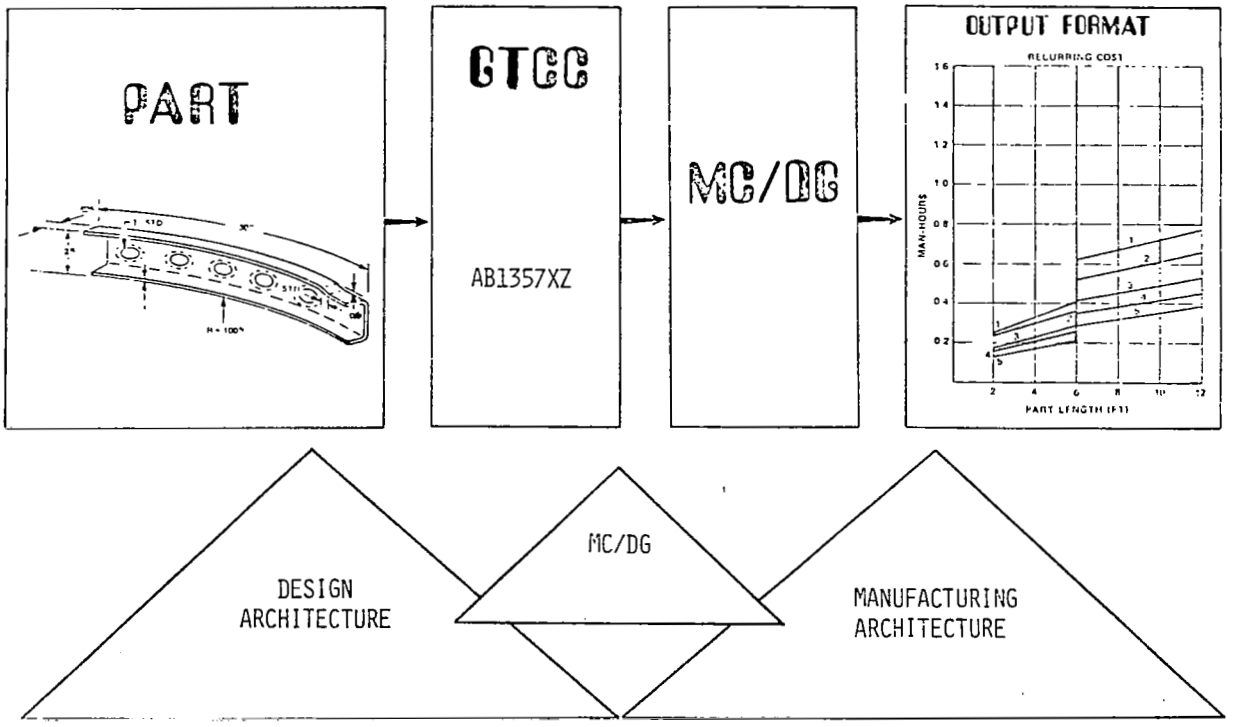


Figure 5.- Manufacturing Cost/Design Guide computerization.
(1 in. = 2.54 cm; 1 ft. = 0.3 m.)

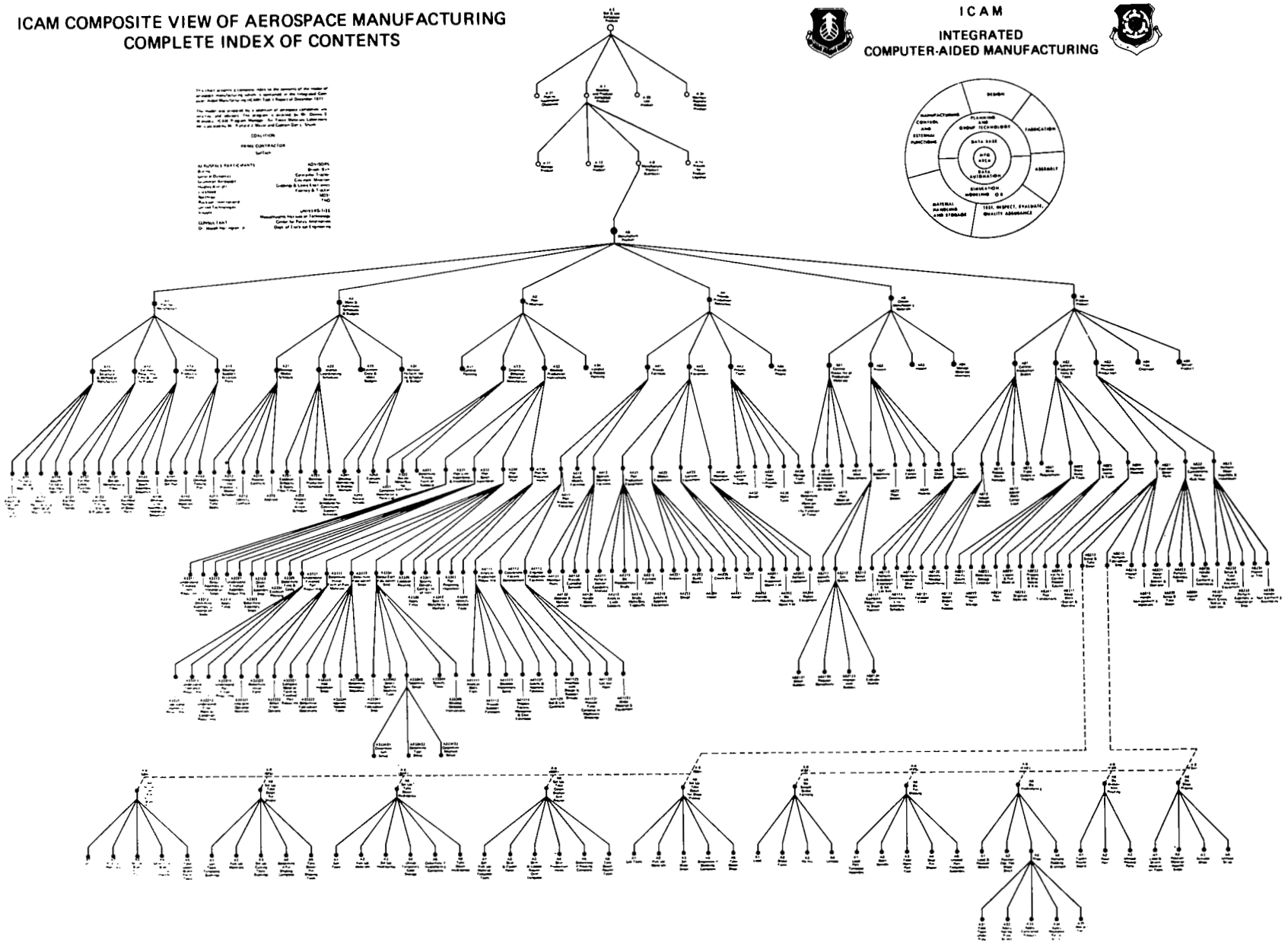


Figure 6.- Manufacturing architecture.

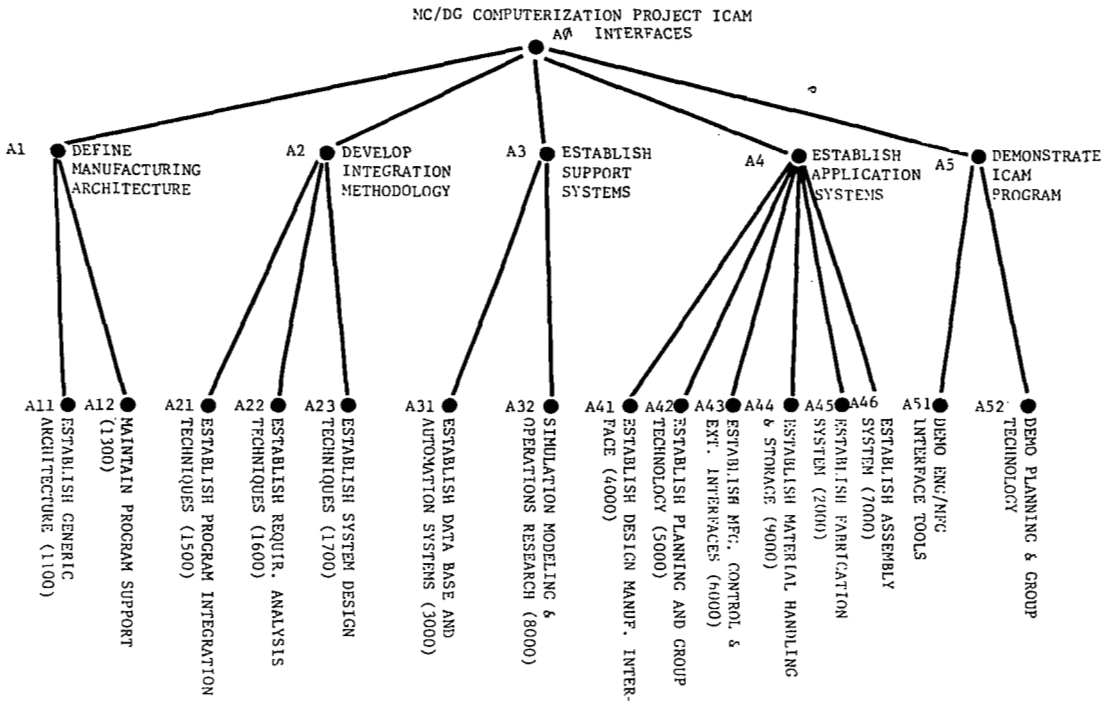


Figure 7.- Subsystem interface.

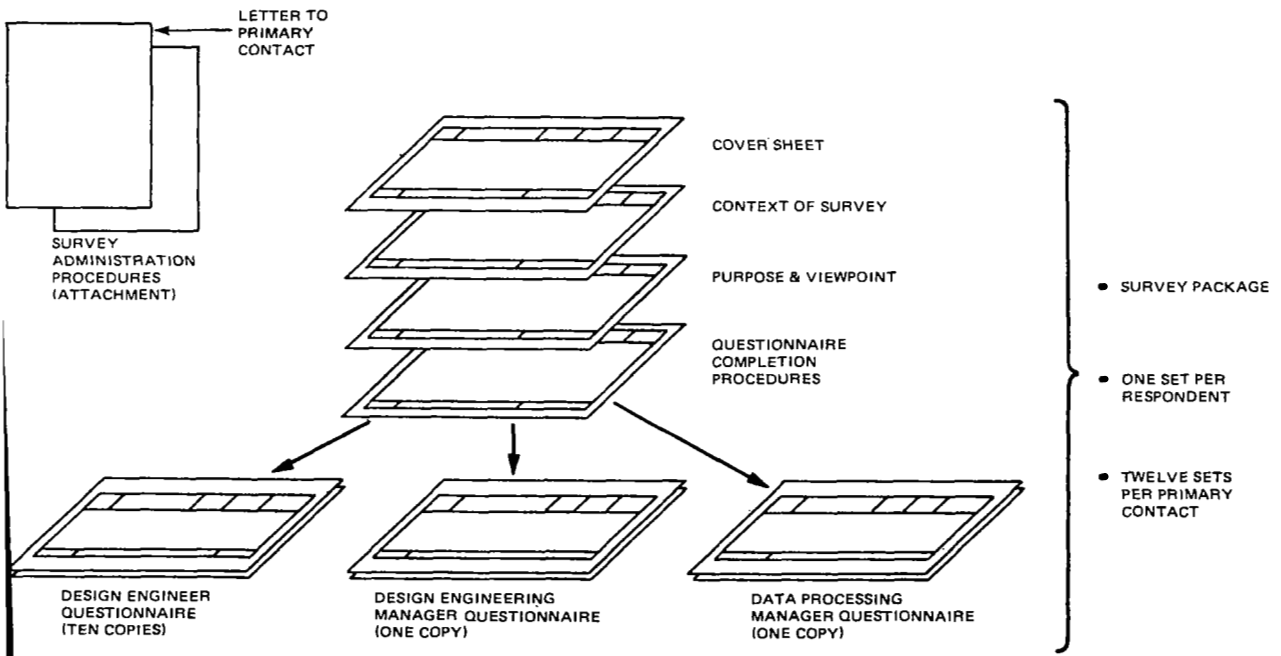


Figure 8.- Survey package.

9. What percentage of your design work comes with cost targets?

a. 100%_____ b. 75%_____ c. 50%_____ d. 25%_____ e. 0%_____

10. If your design work comes with cost targets, what is the source of these targets? (Indicate more than one, if applicable.)

a. Design Engineer (Using Design Procedures, Design Cost, Guide, etc.) _____
b. Conceptual Design Dept. (Using statistical or parametric models) _____
c. Project Management (Based on available funding) _____
d. Corporate Estimating Dept. _____
e. Engineering Support Function (DTC/Value Engineering) _____
f. Other (specify): _____ _____

11. What percentage of your design assignments are defined with specific schedule/time constraints?

a 100%_____ b. 75%_____ c. 50%_____ d. 25%_____ e. 0%_____

12. In what percentage of your design assignments do you perform manufacturing cost design trades?

a. 100%_____ b. 75%_____ c. 50%_____ d. 25%_____ e. 0%_____

13. When you do not perform a manufacturing cost design trade for a design assignment, it usually is because: (rank with 1 = most frequent. Use NA if choice does not apply.)

a. Schedule does not allow adequate time _____
b. No management directive (not the common procedure) _____
c. Special assignment (cost trade not necessary) _____
d. Lack of appropriate data _____
e. Performed by others _____
f. Don't know how _____

Figure 9.- Sample survey page - design engineer questionnaire.

1. To what extent are you involved in the ICAM Program?

a. Contractually involved _____
b. Keep up to date (conferences, reading) _____
c. Heard about it _____
d. No knowledge or involvement _____

2. What percentage of your company's contracts are:

a. Prime _____ b. Subcontracts _____

3. To what extent is your company interested in carrying out computer-aided manufacturing cost/design trade-offs?

a. Extremely interested _____
b. Interested _____
c. Not under consideration _____
d. Definitely not interested _____
e. If not interested, please state reasons: _____

4. Do you currently impose cost targets or guidelines on design engineers?

a. Always _____
b. Usually _____
c. Occasionally _____
d. Rarely _____
e. Never _____

5. Were you familiar with the concept of Manufacturing Cost/Design Guide (MC/DG) before you received this questionna

a. Yes ____ b. No ____

Figure 10.- Sample survey page - design engineering manager questionnaire.

1. What level of experience does your company have in installing/maintaining software systems with on-line interactive applications?

 a. Very experienced____ b. Somewhat experienced____ c. Very little experience____ d. No experience____

2. What levels of experience does your company have in installing/maintaining software systems with graphics-intense applications?

 a. Very experienced____ b. Somewhat experienced____ c. Very little experience____ d. No experience____

3. What types of experience does your company have in computer-aided manufacturing/design support? (Indicate more than one if applicable)

 a. Have developed and maintained our own systems ____
 b. Have acquired externally developed systems ____
 c. Have investigated the use of such systems ____
 d. Have no active experience ____

4. Does your company use CADAM (Computer Aided Design and Manufacturing) or CADAM-like software systems in support of Design and/or Manufacturing activities?

 a. Yes____ b. No____ c. Don't know____

5. Indicate the degree of difficulty you envision in supporting an automated MC/DG.

	1. Extremely Difficult	2. Somewhat Difficult	3. No Problem
a. Maintain the software			
b. Train users			
c. Procure/configure equipment			
d. Monitor usage of system			
e. Provide increased computer operations staff			

Figure 11.- Sample survey page - data processing manager questionnaire.

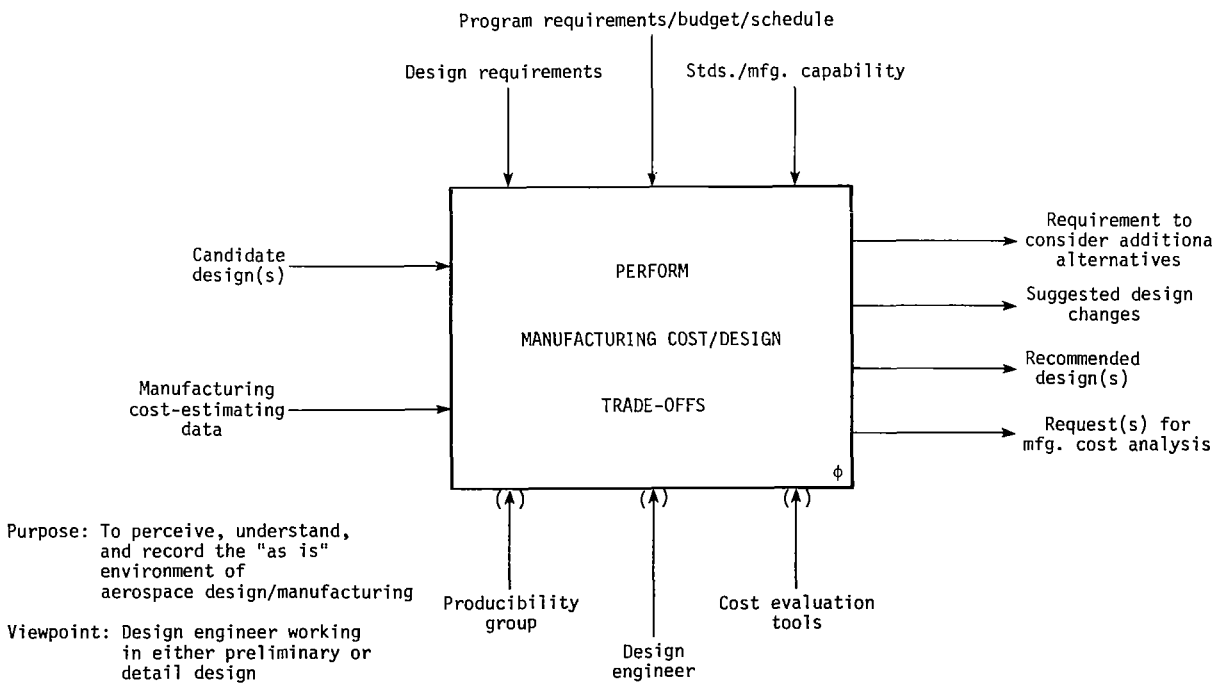


Figure 12.- Example of Composite View IDEF model.

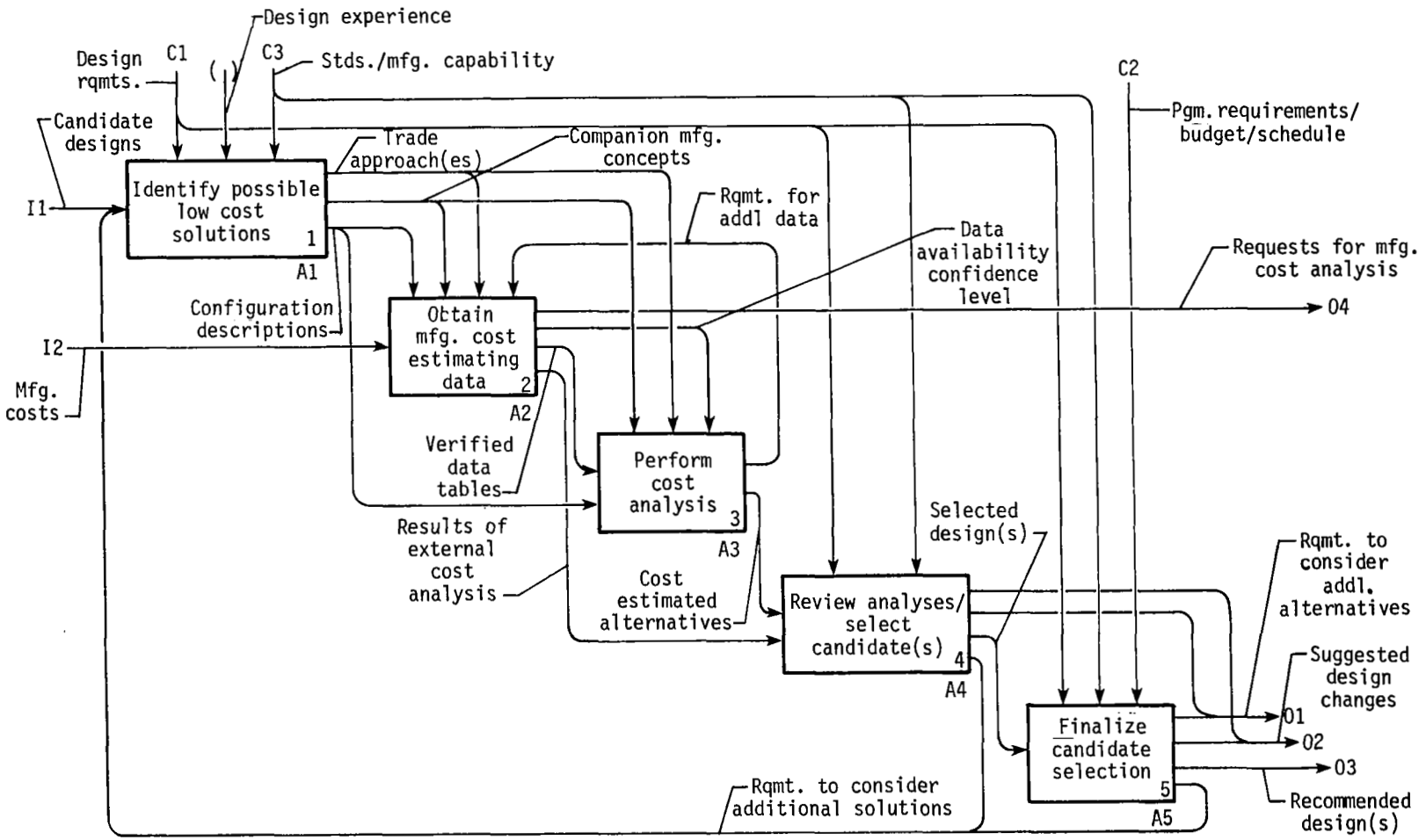


Figure 13.- Detail of model example.

TURNKEY CAD/CAM SYSTEMS' INTEGRATION WITH IPAD SYSTEMS

Robert E. Blauth
Computervision Corporation

ABSTRACT

Today's commercially available turnkey CAD/CAM systems provide a highly interactive environment, and support many specialized application functions for the design/drafting/manufacturing process. This paper presents an overview of several aerospace companies which have successfully integrated turnkey CAD/CAM systems with their own company-wide engineering and manufacturing systems. It also includes a vendor's view of the benefits as well as the disadvantages of such integration efforts. Specific emphasis is placed upon the selection of standards for representing geometric-engineering data and for communicating such information between different CAD/CAM systems.

INTRODUCTION

During the last decade, a number of turnkey Computer Aided Design and Computer Aided Manufacturing (CAD/CAM) systems have evolved which provide highly developed and cost-effective tools for increasing industrial productivity. They support many specialized application functions which automate the engineering design, drafting, and manufacturing processes. One of the largest users of these turnkey systems today is the aerospace industry. Some of the applications in the design and manufacture of airplane products which are supported by the various turnkey CAD/CAM systems are listed below:

- Geometric definition of aircraft surfaces
- Structural layout of internal aircraft parts
- Calculation of mass properties such as weight and center of gravity
- Generation of finite element mesh data for structural analysis
- 3D kinematic visualization
- Layout of fuel and hydraulic tubing systems
- Cockpit instrumentation layout
- Layout of passenger facilities
- Wire harness design and fabrication

Antenna pattern design
Cockpit visibility studies
Creation of layout, detail and assembly drawings
Generation of operation sheets and process planning drawings
Technical publications for maintenance manuals
Generation of N/C toolpath data for drilling, milling, routing and turning operations
3D flat pattern unfolding
Nesting of flat patterns for flame-cutting
Tool and fixture design
Derivation of input for computerized inspection systems

Throughout the next decade, turnkey CAD/CAM systems will play an increasingly important role in the design, engineering, and manufacture of aerospace vehicles. During the next few years, however, turnkey systems will tend to continue to address specific segments of the product development process rather than becoming systems which totally integrate all of a company's CAD/CAM technologies. Thus many aerospace companies, as well as the industry as a whole, have recognized the need to address the integration of their various CAD/CAM activities themselves.

INTEGRATION OF TURNKEY CAD/CAM SYSTEMS WITH AEROSPACE SYSTEMS

By the second half of the 1970's, the benefits of using turnkey CAD/CAM systems and their contributions to improving the overall development cycle for aerospace vehicles were well recognized by the aircraft industry. Many companies such as Boeing, General Dynamics, Grumman, Lockheed, McDonnell Douglas, Northrop, and Rockwell began to use turnkey systems as part of major new engineering development programs. Having realized the need to integrate these turnkey systems with their own existing in-house CAD/CAM facilities, several of the above companies began to develop their own integrated systems.

Grumman Aerospace Corporation

One of the early efforts to integrate a turnkey system with a computerized air vehicle design system was accomplished by Grumman Aerospace Corporation (ref. 1). They developed a system called GEMS (Grumman Engineering and Manufacturing System) which utilized the Lockheed CADAM system for design, drafting, and N/C together with their own software for configuration contour development, structural design and modeling of 2D and 3D space frames, and master dimensioning for sectionalized conic surface generation of aerospace vehicles. A block diagram of the GEMS system is shown in figure 1.

Several design/drafting projects in which the GEMS system was used included the structural drawings for the Louvered Scarfed Shroud System, an infra-red suppressor, and the pod adaptor structure for the F-14 TARPS (Target Acquisition and Reconnaissance Pod System). N/C machining of an EF-111 bulkhead and longeron were also accomplished using GEMS.

Current plans at Grumman include extending GEMS by developing interfaces to other internally developed corporate systems such as RAVES (Rapid Aerospace Vehicle Evaluation System) and General Engineering Programs, as well as to another turnkey CAD/CAM system which they subsequently had acquired from Computervision Corporation.

Northrop Corporation

Northrop Corporation is another example of a company which has developed an integrated CAD/CAM facility by interfacing several turnkey systems with its own internal capabilities (ref. 2). Northrop's system was used to build the main structural box skin for the vertical tail of the F-18 aircraft. Generation of loft data, 3D layouts, 2D detail and assembly drawings, finite element modeling data for structural analysis, tool and fixture design, interactive N/C toolpath generation, and quality assurance data are all supported within the Northrop system.

Boeing Commercial Airplane Company

One of the most ambitious and more recently publicized (ref. 3) programs to integrate all aspects of CAD and CAM for the product definition, analysis, and manufacture of airplane products has been accomplished by the Boeing Commercial Airplane Company.

Boeing has developed a system called CIIN (CAD/CAM Integrated Information Network) which is currently being used to support the development of two new airplanes, the 757 and the 767. The CIIN system integrates: (1) several different turnkey design systems including Computervision, Gerber, and AD-2000, (2) Boeing's internally developed Batch CAD and Interactive Design Analysis systems, and (3) other systems for plotting and APT processing. Figure 2 shows the CIIN system's configuration. The various systems which comprise the network are geographically distributed both in terms of processors, terminals, and resident software. The key to the CIIN system, however, is that a standard form of geometry resides on a centrally located mainframe computer which is accessed and shared by all users and terminals.

The current CIIN system primarily serves to provide interactive design tools to Boeing's engineers for CAD and CAM applications. Future developments will include the integration of the planning and design retrieval activities within Boeing with the existing CAD/CAM applications.

IPAD

The most significant development currently underway to integrate CAD/CAM software and hardware technology for company-wide management of engineering data is the IPAD (Integrated Program for Aerospace-Vehicle Design) program, sponsored by NASA. The integration of turnkey CAD/CAM systems with IPAD systems will ultimately provide the most capability and flexibility to users of such systems.

BENEFITS OF INTEGRATION WITH TURNKEY CAD/CAM SYSTEMS

Several major benefits are available to those companies which have already accomplished some degree of CAD/CAM system integration. First, they can exploit the specialized features and capabilities of a given turnkey system instead of having to re-invent them. Second, they don't necessarily have to lock themselves in to one turnkey vendor if they can develop the appropriate database interfaces within their integrated system. As such, they have the ability to utilize the unique strengths of different turnkey systems for different applications within their development process. For example, one system (A) might be used

for 3D design purposes, a different system (B) used for creating finite element mesh data, and even a third system (C) could be used for generating the detail drawings. Thus, the user of such an integrated system has the flexibility to choose between developing certain functions internally, using a turnkey system which already has such capabilities, or choosing different turnkey systems which each provide unique application features.

An additional benefit is derived by the integrated user in that geometric data can be transmitted to and from the company's subcontractors if they also are users of turnkey CAD/CAM systems. Use of this geometric database can dramatically reduce the errors inherent in interpreting (and often misinterpreting) drawings, as well as the time often spent in re-creating the engineering data, e.g. for manufacturing engineering purposes.

PROBLEMS OF INTEGRATION WITH TURNKEY CAD/CAM SYSTEMS

Three major problems arise for any company which does develop or acquire an integrated system today: maintenance/service, training, and unique sub-system functionality. Once any system becomes composed of different hardware and/or software from different external (or even internal) suppliers, the question of responsibility for maintenance and service becomes important. Clear-cut responsibilities must be assigned in a top-down fashion for maintaining each "module" within the system. Interface standards and test procedures must be completely defined so that problems can be quickly isolated, and "finger-pointing" can be thereby avoided.

Training is another problem which will occur when different systems are integrated. Most turnkey CAD/CAM systems today have their own unique command languages, operating procedures, and hardware input/output devices. Although "standardization" is not possible in these areas in the short term, a concerted effort to make these systems friendlier, i.e. easier to use, is appropriate. If turnkey systems are to become part of IPAD or industry-developed integrated systems, they must provide much simpler user interfaces and more self-teaching aids than they do currently.

Finally, loss of specific sub-system functionality is going to be the largest problem within integrated CAD/CAM systems. Many of the unique CAD/CAM features developed by users (e.g. Grumman, Northrop, Boeing, etc.), supplied by vendors (e.g. Computervision, Gerber, Lockheed, etc.), and supported by integrators (e.g. IPAD) can't be fully exploited on any integrated system today. The reason for this is that no database system currently exists for CAD/CAM applications that can support all the features of every such system. For example, in Boeing's CIIN system described earlier, the "standard form" of geometry completely defines the 3D model data of a part. However, many of the relationships between the model and its associated drawings, as well as other non-geometric information, are lost when the data from a turnkey system is converted to CIIN format and then re-converted back to the same turnkey system. Thus, in fact, the inherent functionality of any integrated system may become the "lowest common denominator" of the functionalities of each of its sub-systems.

TURNKEY CAD/CAM SYSTEMS' INTEGRATION WITH IPAD SYSTEMS

In order to achieve the maximum benefits from an integration of turnkey systems with IPAD, two areas must be addressed. First, it has recently been recognized by the ANSI Y14.26 subcommittee, that a multi-levelled standard should exist for geometric data representation which includes data formats similar to those in use by today's turnkey systems, together with formats for more sophisticated geometric models. As such, the Initial Graphics Exchange Specification (IGES) (ref. 4) prepared by the NBS together with both CAD/CAM users and vendors has been accepted as Sections 2, 3, and 4 of the new ANSI Y14.26.X-198X (ref. 5) proposed standard. IPAD's support of this new ANSI standard will greatly facilitate the integration with turnkey systems. In addition, the current IPAD program has focused primarily on the design, drafting, and N/C aspects of aerospace-vehicle development. In the future, IPAD should consider extending its geometry format, which is currently mechanical design oriented, to include electrical and piping applications as well.

Secondly, interface standards for communication protocols must be defined to efficiently support communication between IPAD systems and turnkey CAD/CAM systems. Standards should be established for disk, cassette, and magnetic tape formats, as well as for direct data communication links between such systems. A cooperative effort, using the collective experience of the turnkey system vendors, users, and IPAD personnel, would help to effect the development of compatible interfaces between IPAD and non-IPAD systems.

SUMMARY

During the next decade, turnkey CAD/CAM system vendors will invest between $500 million and $1 billion dollars in developing new and improved CAD/CAM system capabilities. The ability for IPAD users to interface their systems with turnkey CAD/CAM systems will provide the basis for major improvements in industrial productivity throughout the 1980's. The key to accomplishing this integration is through the use of standard file structures and standard communication formats for transmitting geometric, engineering and manufacturing data between different systems. CAD/CAM is a dynamic technology for aerospace applications in the 1980's. In order to remain competitive, these companies will have to be able to combine turnkey, IPAD, and in-house facilities in a totally integrated system.

REFERENCES

1. Rosenbaum, J.D.: Engineering Systems for Air Vehicle Design at the Grumman Aerospace Corporation. Engineering Society of Detroit Computer Graphics Conference, April 1978.

2. Allen, D.C.: Techniques: Computer-Integrated Manufacturing. Datamation, vol. 26, no. 3, March 1980, pp. 187-190.

3. Jones, J.C.: Lessons Learned in Developing CAD at Boeing. Deere & Company CAD/CAM Conference Proceedings, January 1980, pp. 97-106.

4. Initial Graphics Exchange Specification, Version 1.0. National Bureau of Standards (report number NBSIR 80-1978 (R)).

5. Digital Representation of Physical Object Shapes. ANSI Y14.26.X-198X, June 1980.

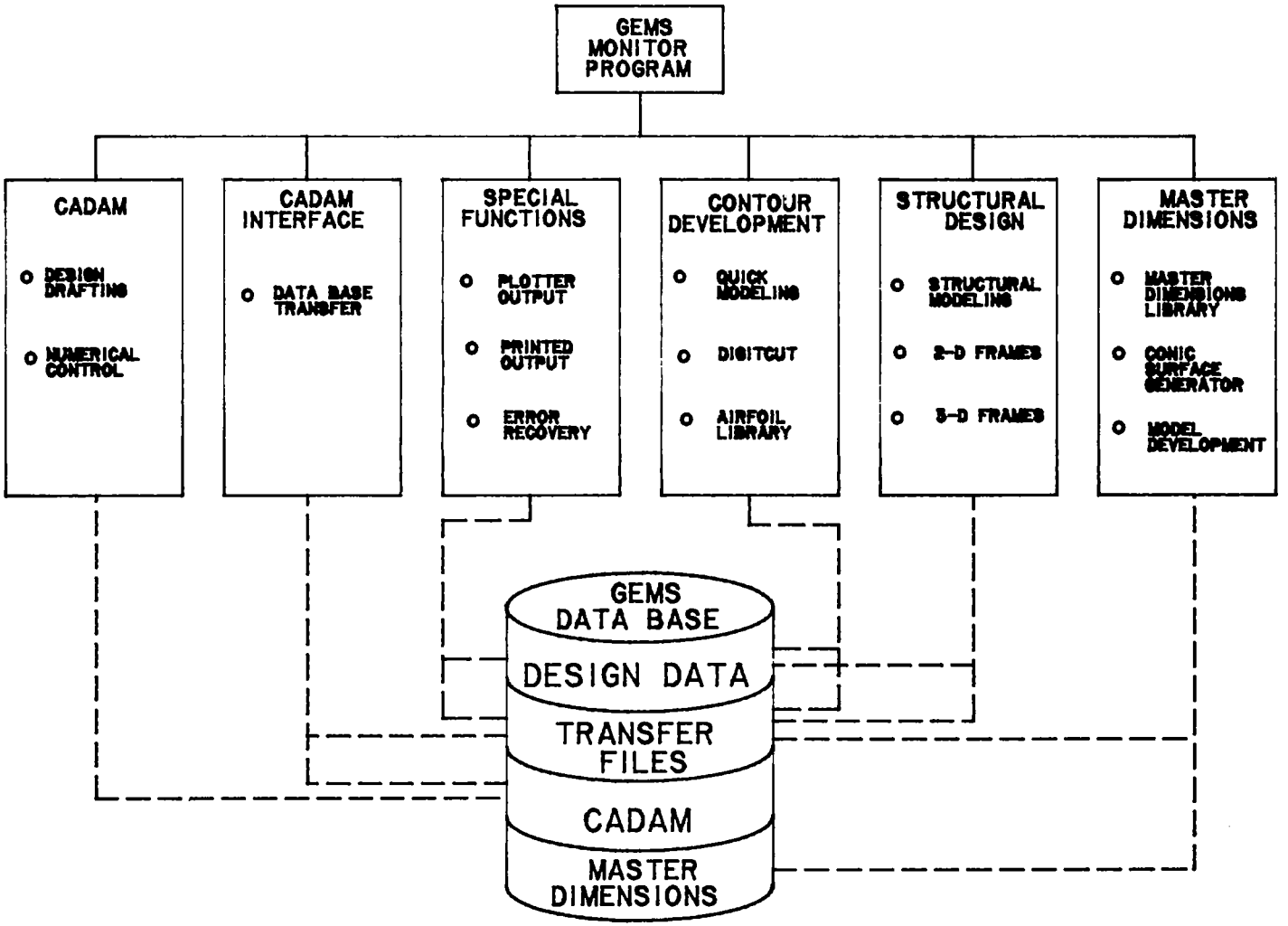


Figure 1.- Grumman's GEMS system.

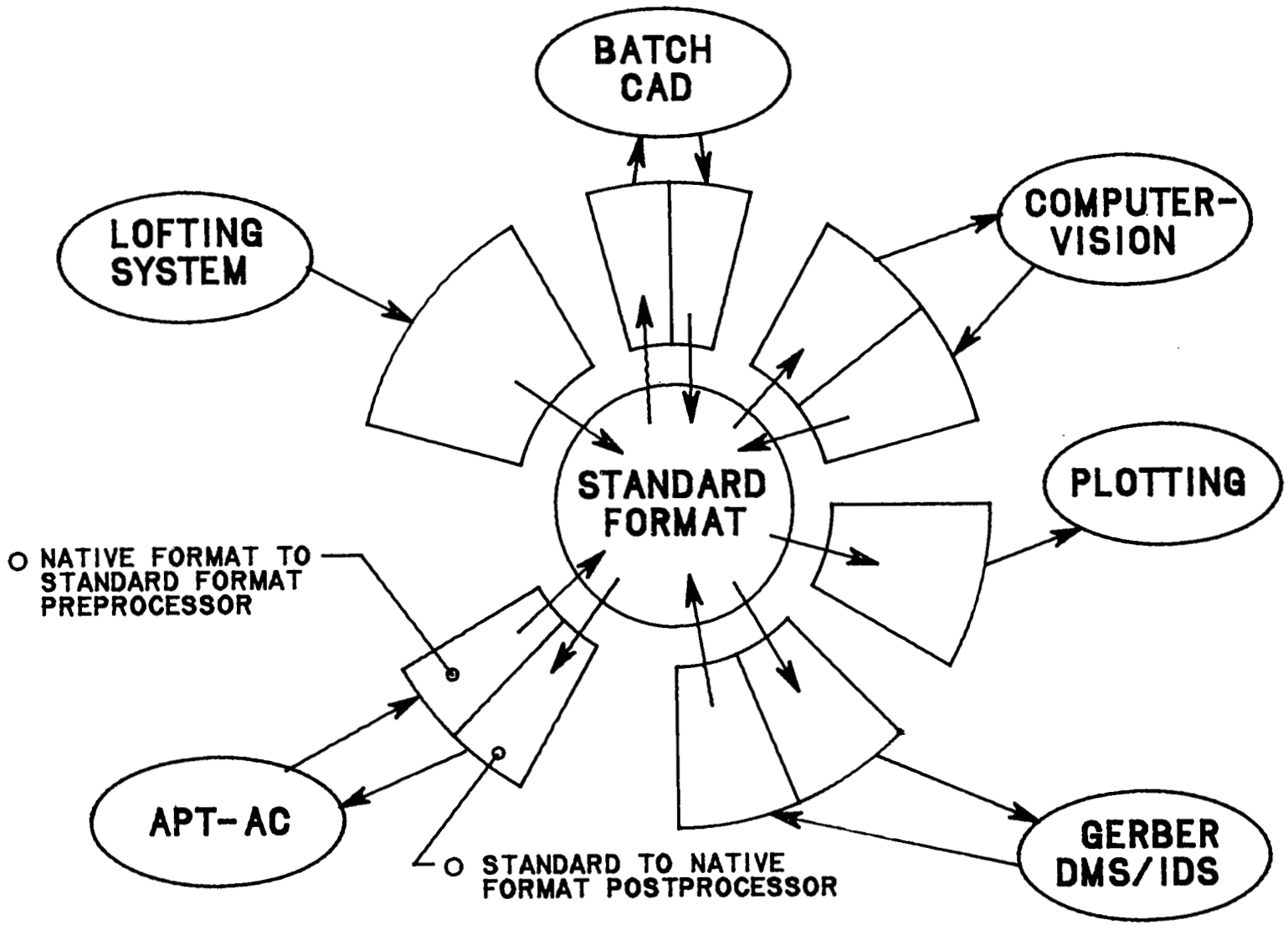


Figure 2.- Boeing's CIIN configuration.

GLOSSARY

ACT
: Acceptance Criteria for Testing

AD 2000
: A commercially available product of Manufacturing and Consulting Services for computerized design drafting/N.C. programming functions.

AGDM
: Agent Directory Manager

ANSI
: American National Standards Institute

ANSI/SPARK THREE SCHEMA DATA MODEL
: A data model described in an ACM SIGMOD FDT Bulletin which first recognized the need for external, conceptual, and internal data base descriptions.

AP
: Access Primitives

APIL
: Application Program Interface Language

APT
: Automatically Programmed Tools - A manufacturing application computer program.

APU
: Auxiliary Power Unit

ART
: Average Response Time

ATTRIBUTE
: Representation of the use of a domain within a relation. An attribute can be thought of as a column in a table whose values are drawn from a domain.

B-TREE
: Hierarchical indexing method that always maintains a balance between the number of entries in the branches of the hierarchy.

BAL
: Basic Assembly Language

BATCH
: A group of jobs to be run on a computer at one time with the same program.

BCD
: Binary Coded Decimal

CAD
Computer Aided Design

CADAM
Computer-Graphics Augmented Design and Manufacturing (Lockheed System)

CAM
Computer Aided Manufacturing

CCL
Conceptual Command Language - Language used in IPIP to perform operations on tuples at the conceptual level record level.

CCLD
Conceptual Command Language Driver

CCLPR
Conceptual Command Language Processing Routines

CCP
Conceptual Constraint Processor

CCR
Configuration Change Request

CDR
Critical Design Review

CHUNK
An unstructured data set that exists outside the bounds of the formally defined data bases. It has no definition of format, nor does IPIP know what data elements are in it.

CIT
Conceptual Internal Translator

CITRTS
Conceptual-to-Internal Translator Run-Time Subsystem

CITT
Conceptual Internal Tuple Translator

CNC
Computer Numerical Control

COCAM
Committee on Computer Aided Manufacturing

CODASYL
Conference on Data System Languages - a group of computer vendors, users, and government agencies formed in 1959 to recommend a common high-level language.

CODASYL DBTG MODEL
A network data base model. The fundamental data structure is the record, and records are related into logical structures using the notion of sets.

CONCEPTUAL COMMAND LANGUAGE
Language used in IPIP for performing operations on tuples at the conceptual record level. This is the means by which the external-to-conceptual run-time subsystem communicates with the conceptual-to-internal run-time subsystem.

CONCEPTUAL SCHEMA
A description of all data known to the data management system and independent of any user views (external schemas) or physical storage considerations (internal schemas). The conceptual schema is compiled into tables and used by the external schema compiler, the internal schema compiler, and the run-time subsystems.

CONCEPTUAL SCHEMA COMPILER
The component of IPIP that compiles a conceptual schema source and produces conceptual schema object tables which are stored by DIMS.

CONCEPTUAL-TO-INTERNAL RUN-TIME SUBSYSTEM
A set of routines and processors within IPIP using the tables set up by the conceptual-to-internal translator to perform the transformations between conceptual schema records and internal schema records. Also processes keys by using and maintaining the files created for keyed attributes.

CONSTRAINT
A condition or set of conditions specified by the conceptual schema author for establishing rules of integrity that must be satisfied by data within the data base, including range checks, uniqueness requirements, and subset conditions.

CPU
Central Processing Unit

CR
Conceptual Relation

CRI
Conceptual Relation Instance

CRT
Cathode Ray Tube

CRV
Curve

CS
Conceptual Schema - A description of all data known to the data management system. It is independent of the users' view or the physical devices.

CSA
Common Service Area

CSC
Conceptual Schema Compiler

CSN
Conceptual Schema Name

CSS
Conceptual Schema Source

CT
Conceptual Transaction

DATA AREA
A logical grouping of data sets based on configuration control permissions of the data set. Data areas relate to each other in a hierarchical manner that generally follows the structure of the user's organization.

DATA BASE
A data base is all user data logically associated with a single conceptual schema as well as its associated undefined data sets.

DATA DEFINITION LANGUAGE
Language used by the data base administrator to describe the content of the conceptual, external, and internal schemas.

DATA MANIPULATION LANGUAGE
Data manipulation language used by the application programmers to manipulate data in the data base from within an application. There is a unique data manipulation language for each type of external schema supported by the data base management system.

DATA SET
An arbitrary collection of records as grouped by the user. A data set may be a defined data set (i.e., as defined by the three schemas) or an undefined data set.

DBA
Data Base Administrator - An individual or organization having responsibility for data bases.

DBMS
Data Base Management System - A generalized software system for managing data bases and making interrogation, maintenance and analysis of data available to users.

DCS
Distributed Computing System - A network of computer systems whose control is usually distributed throughout the system.

DDL
Data Definition Language - A language for defining a data model together with part of its mapping to storage.

DDS
Data Definition Subsystem

DOMAIN
A universal set of values from which the actual values appearing in a column (attribute) of a table (relation) are drawn.

DPT
Distributed Processing Table

DRTS
Distributed Run-Time Subsystem

DS
Data Set

ECM
Executive Command Mode

ES
External Schema - A description, prepared by a data base administrator, of the user's view of the data and data relationship contained in the conceptual schema.

ESC
External Schema Compiler

ESN
External Schema Name

ESO
External Schema Object

ESRN
External Schema Record Name

ESS
External Schema Source

EST
External Schema Translator

EXTERNAL RUN-TIME SUBSYSTEMS
Subsystems that process run-time requests from user application programs and query users to the internal run-time subsystem via the conceptual command language (CCL).

EXTERNAL SCHEMA

A description of the user's view of data and data relationship contained in the conceptual schema. The data base administrator writes an external schema for a particular user view describing types of data the user can view and relationships existing within that data.

EXTERNAL SCHEMA COMPILER

The component that converts a user's source external view of the data base into a set of object tables used by the external-to-conceptual translator. One compiler is required for each different external data model used (relational, network, geometric, etc.).

EXTERNAL-TO-CONCEPTUAL SCHEMA TRANSLATOR

A processor which maps external records into conceptual records, and vice versa, for the purpose of satisfying user requests upon IPIP.

FEDD

For Early Domestic Dissemination - A constraint concerning foreign distribution placed upon most IPAD documentation.

GEOMETRIC EXTERNAL SCHEMA

An external schema written in a geometry data definition language describing the structure of engineering objects.

GPGS

General Purpose Graphics System

GPGS-F

General Purpose Graphics System - A commercially available product of University of Trondheim, Norway, which is a FORTRAN library of graphics subroutines.

GRTS

Geometry External-to-Conceptual Translator Run-Time Subsystem

HEADER DATA

Specific data stored in the DIMS data bank consisting of data set name, version, owner ID, creation date and time, security, retention information, processing histories, etc.

HLGR

Higher-Level Graphics Related Software

HYPERCHANNEL

A commercially available product of Network Systems Corporation which is a high speed data bus capable of transfer rates of 50 megabits.

ICAM

Integrated Computer-Aided Manufacturing - A large CAM system being developed with Air Force support and industry cooperation.

IDR
Intermediate Design Review

IF
IPEX Function

INF
IPIP Navigational Facility

INFORMATION BANK
The total accumulation of data known to IPAD, consisting of data within the data base as well as chunks. (Data is considered to be known to IPAD if there is header data in the IPAD system bearing information about the data.) The information bank is complete and self-describing.

INTERNAL RUN-TIME SUBSYSTEM
A set of routines and processors within IPIP that use the tables set up by the conceptual-to-internal translator to actually perform the data transfers between conceptual schema records and internal schema records. Also processes keys by using and maintaining the files created for keyed attributes.

INTERNAL SCHEMA
A tabular description of how the data corresponding to a conceptual schema physically resides on the mass storage devices. It describes to the data management system what data fields are to be treated as keys, how large the physical records are, as well as other details needed by the run-time system to store and retrieve data from the devices.

INTERNAL SCHEMA COMPILER
A translator that produces an internal schema object (ISO) consisting of tables containing essential data items from the internal schema source (ISS). These tables are stored in the metadata base by the data inventory management subsystem (DIMS). The compiler checks syntax and performs validation checks with conceptual schema information in the metadata base. Output also includes diagnostics and listings of the ISS.

I/O
Input/Output

IPAD
Integrated Programs for Aerospace Vehicle Design

IPEX
IPAD Executive

IPIP
IPAD Information Processor - The data base management system of IPAD.

IPIP PRECOMPILERS
Programs that process application programs containing embedded data manipulation commands that are translated into source language communication areas and subroutine calls for communication with the IPIP run-time subsystems.

IPO
IPAD Project Office (NASA Langley Research Center)

IPSR
IPEX Service Routines

IRBN
Internal Relative Block Numbers

IRQL
IPIP Relational Query Language

ISC
Internal Schema Compiler

ISO
Internal Schema Object

ISS
Internal Schema Source

ITAB
Industry Technical Advisory Board - A group giving advice and guidance to the IPAD development.

JEQ
Job Execution Queue

KEY
For conceptual or external schemas, an attribute or combination of attributes that uniquely identify the tuples of a relation. For internal schemas, a key is one or more attributes having a special access structure, such as B-trees or hashing.

LDB
Local Data Base

LNCP
Local Network Control Program

MCP
Message Control Point

MENS
Mission Element Needs Statement

METADATA BASE
An instance of a DIMS conceptual schema including the tables produced by the schema compilers, precompilers, and translators.

MIF
Migration Interface Feature

MSGRTNG
Message Routing

NA
Network Adapter

NAVIGATIONAL DATA MODEL
A data model representing data as a collection of records where relationships between records which are known to the data management system are represented in terms of name connectors called non-information bearing sets. The navigational data model is a subset of the CODASYL DBTG model.

NC
Numerical Control - Control of machines by computer output.

NERTS
Navigational External-to-Conceptual Translator Run-Time Subsystem

NOS
Network Operating System (Control Data Corporation)

NPC
Navigational Pre-Compiler

OAST
Office of Science and Technology (NASA)

OM
Operating Module

OS
Operating System

OSI
Operating Systems Interface

PCC
Program Control Center

PDB
Process Distributor Block

PDF
Pseudo Display File

PDR
Preliminary Design Review

PE
Processing Element

PIN
Program Item Number

PMD
Post Mortem Dump

PNR
Partitioned With No Redundancy

PS
Pre-Compiler Subsystem

PWR
Partitioned With Redundancy

QLS
Query Language Subsystem

QRTS
Query External-to-Conceptual Translator Run-Time Subsystem

QUERY LANGUAGE
A self-contained language allowing users access to an IPAD data base using relational external schemas to represent the user's view of the data base.

RELATION
A table where each column heading (or domain) is a member of the collection of sets D_1, D_2, ..., D_n and each row is member of the set of n-tuples or Cartesian product on D_1, D_2, ..., D_n. Given a collection of sets D_1, D_2, ..., D_n (not necessarily distinct), R is a relation of these n sets if it is a set of n-tuples (d_1, d_2, ..., d_n) such that d_1 belongs to D_1, d_2 belongs to D_2, ..., d_n belongs to D_n.

RELATIONAL DATA MODEL
A data model representing data as a collection of relations where relationships between relations are represented in terms of comparable fields. (There are no predefined relationships - these are carried in the data.)

RELATIONAL INFORMATION MANAGER
A prototype data base management system based on the relational data model.

RIMS
Relational Information Management System - A prototype of a proposed IPAD data management system.

RP
Request Processes

RUN-TIME SUBSYSTEMS
Components of a data base management system invoked at the time of execution of process user DML commands.

SAMM
Systematic Activity Modeling Method (Boeing Computer Services)

SCHEMA
A description or definition of a data base

SCL
Scalar

SGA
Shared Global Area

SM
Source Module

SP
Server Processes

SPR
Software Problem Report

TCP
Transport Control Program

TDF
Transformed Display File

TIGS
Terminal Interactive Graphics Systems

TPE
Technical Program Element

TUPLE
A row of a relation, also referred to as an n-tuple. In IPAD documents the terms "n-tuple" and "record" are used interchangeably.

UCC
Utility Command and Control

UF
 User Function

UI
 User Interface

VDB
 Versioned Data Base

VDBP
 Versioned Data Base Processor

VMCF
 Vertical Machine Communication Facility

WBS
 Work Breakdown Structure

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CP-2143		

4. Title and Subtitle	5. Report Date
IPAD: INTEGRATED PROGRAMS FOR AEROSPACE-VEHICLE DESIGN	September 1980
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No. L-13916
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665	10. Work Unit No. 510-54-13-01
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546 and Industry Technical Advisory Board c/o Boeing Commercial Aircraft Company Seattle, WA 98124	13. Type of Report and Period Covered Conference Publication
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The conference was organized to promote wider awareness of the IPAD program and its coming impact on American industry. The program focuses on technology issues that are critical to computer-aided design and manufacturing. Included is a description of a representative aerospace design process and its interface with manufacturing, the design of a future IPAD integrated computer-aided design system, results to date in developing IPAD products and associated technology, and industry experiences and plans to exploit these products.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Computer-aided design (CAD) Computer-aided manufacturing (CAM)	FEDD Distribution Subject Category 62

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	396	

Available: NASA's Industrial Application Centers

NASA-Langley, 1980

Industrial Applications Centers

Aerospace Research Applications Center
1201 East 38th Street
Indianapolis, Indiana 46205

E. G. Buck, director
Phone: (317) 264-4644

Kerr Industrial Applications Center
Southeastern Oklahoma State University
Durant, Oklahoma 74701

Robert Oliver, Ph.D., director
Phone: (405) 924-0121, Ext. 413

NASA Industrial Applications Center
701 LIS Building
University of Pittsburgh
Pittsburgh, Pennsylvania 15260

Paul A. McWilliams, Ph.D., executive director
Phone: (412) 624-5211

NASA Industrial Applications Center
University of Southern California
Denny Research Building
University Park
Los Angeles, California 90007

Robert Mixer, Ph.D., associate director
Phone: (213) 741-6132

New England Research Applications Center
Mansfield Professional Park
Storrs, Connecticut 06268

Daniel Wilde, Ph.D., director
Phone: (203) 486-4533

North Carolina Science and Technology Research Center
Post Office Box 12235
Research Triangle Park, North Carolina 27709

Peter J. Chenery, director
Phone: (919) 549-0671

Technology Applications Center (TAC)
University of New Mexico
2500 Central Avenue, S.E.
Albuquerque, New Mexico 87131

Stanley Morain, Ph.D., director
Phone: (505) 277-3622